

Introduction

Nix is a *purely functional package manager*. This means that it treats packages like values in purely functional programming languages such as Haskell — they are built by functions that don't have side-effects, and they never change after they have been built. Nix stores packages in the *Nix store*, usually the directory `/nix/store`, where each package has its own unique subdirectory such as

```
/nix/store/b6gvzjyb2pg0kjfwrjmg1vfhh54ad73z-firefox-33.1/
```

where `b6gvzjyb2pg0..` is a unique identifier for the package that captures all its dependencies (it's a cryptographic hash of the package's build dependency graph). This enables many powerful features.

Multiple versions

You can have multiple versions or variants of a package installed at the same time. This is especially important when different applications have dependencies on different versions of the same package — it prevents the “DLL hell”. Because of the hashing scheme, different versions of a package end up in different paths in the Nix store, so they don't interfere with each other.

An important consequence is that operations like upgrading or uninstalling an application cannot break other applications, since these operations never “destructively” update or delete files that are used by other packages.

Complete dependencies

Nix helps you make sure that package dependency specifications are complete. In general, when you're making a package for a package management system like RPM, you have to specify for each package what its dependencies are, but there are no guarantees that this specification is complete. If you forget a dependency, then the package will build and work correctly on *your* machine if you have the dependency installed, but not on the end user's machine if it's not there.

Since Nix on the other hand doesn't install packages in “global” locations like `/usr/bin` but in package-specific directories, the risk of incomplete dependencies is greatly reduced. This is because tools such as compilers don't search in per-packages directories such as `/nix/store/5lbfaxb722zp...-openssl-0.9.8d/include`, so if a package builds correctly on

your system, this is because you specified the dependency explicitly. This takes care of the build-time dependencies.

Once a package is built, runtime dependencies are found by scanning binaries for the hash parts of Nix store paths (such as `r8vvq9kq...`). This sounds risky, but it works extremely well.

Multi-user support

Nix has multi-user support. This means that non-privileged users can securely install software. Each user can have a different *profile*, a set of packages in the Nix store that appear in the user's `PATH`. If a user installs a package that another user has already installed previously, the package won't be built or downloaded a second time. At the same time, it is not possible for one user to inject a Trojan horse into a package that might be used by another user.

Atomic upgrades and rollbacks

Since package management operations never overwrite packages in the Nix store but just add new versions in different paths, they are *atomic*. So during a package upgrade, there is no time window in which the package has some files from the old version and some files from the new version — which would be bad because a program might well crash if it's started during that period.

And since packages aren't overwritten, the old versions are still there after an upgrade. This means that you can *roll back* to the old version:

```
$ nix-env --upgrade --attr nixpkgs.some-package  
$ nix-env --rollback
```

Garbage collection

When you uninstall a package like this...

```
$ nix-env --uninstall firefox
```

the package isn't deleted from the system right away (after all, you might want to do a rollback, or it might be in the profiles of other users). Instead, unused packages can be deleted safely by running the *garbage collector*:

```
$ nix-collect-garbage
```

This deletes all packages that aren't in use by any user profile or by a currently running program.

Functional package language

Packages are built from *Nix expressions*, which is a simple functional language. A Nix expression describes everything that goes into a package build task (a “derivation”): other packages, sources, the build script, environment variables for the build script, etc. Nix tries very hard to ensure that Nix expressions are *deterministic*: building a Nix expression twice should yield the same result.

Because it's a functional language, it's easy to support building variants of a package: turn the Nix expression into a function and call it any number of times with the appropriate arguments. Due to the hashing scheme, variants don't conflict with each other in the Nix store.

Transparent source/binary deployment

Nix expressions generally describe how to build a package from source, so an installation action like

```
$ nix-env --install --attr nixpkgs.firefox
```

could cause quite a bit of build activity, as not only Firefox but also all its dependencies (all the way up to the C library and the compiler) would have to be built, at least if they are not already in the Nix store. This is a *source deployment model*. For most users, building from source is not very pleasant as it takes far too long. However, Nix can automatically skip building from source and instead use a *binary cache*, a web server that provides pre-built binaries. For instance, when asked to build `/nix/store/b6gvzjyb2pg0...-firefox-33.1` from source, Nix would first check if the file `https://cache.nixos.org/b6gvzjyb2pg0...narinfo` exists, and if so, fetch the pre-built binary referenced from there; otherwise, it would fall back to building from source.

Nix Packages collection

We provide a large set of Nix expressions containing hundreds of existing Unix packages, the *Nix Packages collection* (Nixpkgs).

Managing build environments

Nix is extremely useful for developers as it makes it easy to automatically set up the build environment for a package. Given a Nix expression that describes the dependencies of your package, the command `nix-shell` will build or download those dependencies if they're not already in your Nix store, and then start a Bash shell in which all necessary environment variables (such as compiler search paths) are set.

For example, the following command gets all dependencies of the Pan newsreader, as described by [its Nix expression](#):

```
$ nix-shell '<nixpkgs>' --attr pan
```

You're then dropped into a shell where you can edit, build and test the package:

```
[nix-shell]$ unpackPhase  
[nix-shell]$ cd pan-*  
[nix-shell]$ configurePhase  
[nix-shell]$ buildPhase  
[nix-shell]$ ./pan/gui/pan
```

Portability

Nix runs on Linux and macOS.

NixOS

NixOS is a Linux distribution based on Nix. It uses Nix not just for package management but also to manage the system configuration (e.g., to build configuration files in `/etc`). This means, among other things, that it is easy to roll back the entire configuration of the system to an earlier state. Also, users can install software without root privileges. For more information and downloads, see the [NixOS homepage](#).

License

Nix is released under the terms of the [GNU GPLv2.1 or \(at your option\) any later version](#).

Quick Start

This chapter is for impatient people who don't like reading documentation. For more in-depth information you are kindly referred to subsequent chapters.

1. Install Nix by running the following:

```
$ curl -L https://nixos.org/nix/install | sh
```

The install script will use `sudo`, so make sure you have sufficient rights. On Linux, `--daemon` can be omitted for a single-user install.

For other installation methods, see [here](#).

2. See what installable packages are currently available in the channel:

```
$ nix-env --query --available --attr-path
nixpkgs.docbook_xml_dtd_43          docbook-xml-4.3
nixpkgs.docbook_xml_dtd_45          docbook-xml-4.5
nixpkgs.firefox                      firefox-33.0.2
nixpkgs.hello                        hello-2.9
nixpkgs.libxslt                      libxslt-1.1.28
...
...
```

3. Install some packages from the channel:

```
$ nix-env --install --attr nixpkgs.hello
```

This should download pre-built packages; it should not build them locally (if it does, something went wrong).

4. Test that they work:

```
$ which hello
/home/eelco/.nix-profile/bin/hello
$ hello
Hello, world!
```

5. Uninstall a package:

```
$ nix-env --uninstall hello
```

6. You can also test a package without installing it:

```
$ nix-shell --packages hello
```

This builds or downloads GNU Hello and its dependencies, then drops you into a Bash shell where the `hello` command is present, all without affecting your normal environment:

```
[nix-shell:~]$ hello
```

```
Hello, world!
```

```
[nix-shell:~]$ exit
```

```
$ hello
```

```
hello: command not found
```

7. To keep up-to-date with the channel, do:

```
$ nix-channel --update nixpkgs  
$ nix-env --upgrade '*'
```

The latter command will upgrade each installed package for which there is a “newer” version (as determined by comparing the version numbers).

8. If you’re unhappy with the result of a `nix-env` action (e.g., an upgraded package turned out not to work properly), you can go back:

```
$ nix-env --rollback
```

9. You should periodically run the Nix garbage collector to get rid of unused packages, since uninstalls or upgrades don’t actually delete them:

```
$ nix-collect-garbage --delete-old
```

Installation

This section describes how to install and configure Nix for first-time use.

The current recommended option on Linux and MacOS is [multi-user](#).

Multi-user

This installation offers better sharing, improved isolation, and more security over a single user installation.

This option requires either:

- Linux running systemd, with SELinux disabled
- MacOS

```
$ bash <(curl -L https://nixos.org/nix/install) --daemon
```

Single-user

Single-user is not supported on Mac.

This installation has less requirements than the multi-user install, however it cannot offer equivalent sharing, isolation, or security.

This option is suitable for systems without systemd.

```
$ bash <(curl -L https://nixos.org/nix/install) --no-daemon
```

Distributions

The Nix community maintains installers for several distributions.

They can be found in the [nix-community/nix-installers](#) repository.

Supported Platforms

Nix is currently supported on the following platforms:

- Linux (i686, x86_64, aarch64).
- macOS (x86_64, aarch64).

Installing a Binary Distribution

The easiest way to install Nix is to run the following command:

```
$ curl -L https://nixos.org/nix/install | sh
```

This will run the installer interactively (causing it to explain what it is doing more explicitly), and perform the default "type" of install for your platform:

- single-user on Linux
- multi-user on macOS

Notes on read-only filesystem root in macOS 10.15 Catalina +

- It took some time to support this cleanly. You may see posts, examples, and tutorials using obsolete workarounds.
- Supporting it cleanly made macOS installs too complex to qualify as single-user, so this type is no longer supported on macOS.

We recommend the multi-user install if it supports your platform and you can authenticate with `sudo`.

Single User Installation

To explicitly select a single-user installation on your system:

```
$ curl -L https://nixos.org/nix/install | sh -s -- --no-daemon
```

This will perform a single-user installation of Nix, meaning that `/nix` is owned by the invoking user. You can run this under your usual user account or root. The script will invoke `sudo` to create `/nix` if it doesn't already exist. If you don't have `sudo`, you should manually create `/nix` first as root, e.g.:

```
$ mkdir /nix
$ chown alice /nix
```

The install script will modify the first writable file from amongst `.bash_profile`, `.bash_login` and `.profile` to source `~/.nix-profile/etc/profile.d/nix.sh`. You can set

the `NIX_INSTALLER_NO MODIFY PROFILE` environment variable before executing the install script to disable this behaviour.

Multi User Installation

The multi-user Nix installation creates system users, and a system service for the Nix daemon.

Supported Systems

- Linux running systemd, with SELinux disabled
- macOS

You can instruct the installer to perform a multi-user installation on your system:

```
$ curl -L https://nixos.org/nix/install | sh -s -- --daemon
```

The multi-user installation of Nix will create build users between the user IDs 30001 and 30032, and a group with the group ID 30000. You can run this under your usual user account or root. The script will invoke `sudo` as needed.

Note

If you need Nix to use a different group ID or user ID set, you will have to download the tarball manually and [edit the install script](#).

The installer will modify `/etc/bashrc`, and `/etc/zshrc` if they exist. The installer will first back up these files with a `.backup-before-nix` extension. The installer will also create `/etc/profile.d/nix.sh`.

macOS Installation

We believe we have ironed out how to cleanly support the read-only root on modern macOS. New installs will do this automatically.

This section previously detailed the situation, options, and trade-offs, but it now only outlines what the installer does. You don't need to know this to run the installer, but it may help if you run into trouble:

- create a new APFS volume for your Nix store
- update `/etc/synthetic.conf` to direct macOS to create a "synthetic" empty root directory to mount your volume
- specify mount options for the volume in `/etc/fstab`
 - `rw` : read-write
 - `noauto` : prevent the system from auto-mounting the volume (so the LaunchDaemon mentioned below can control mounting it, and to avoid masking problems with that mounting service).
 - `nobrowse` : prevent the Nix Store volume from showing up on your desktop; also keeps Spotlight from spending resources to index this volume
- if you have FileVault enabled
 - generate an encryption password
 - put it in your system Keychain
 - use it to encrypt the volume
- create a system LaunchDaemon to mount this volume early enough in the boot process to avoid problems loading or restoring any programs that need access to your Nix store

Installing a pinned Nix version from a URL

Version-specific installation URLs for all Nix versions since 1.11.16 can be found at releases.nixos.org. The corresponding SHA-256 hash can be found in the directory for the given version.

These install scripts can be used the same as usual:

```
$ curl -L https://releases.nixos.org/nix/nix-<version>/install | sh
```

Installing from a binary tarball

You can also download a binary tarball that contains Nix and all its dependencies. (This is what the install script at <https://nixos.org/nix/install> does automatically.) You should unpack it somewhere (e.g. in `/tmp`), and then run the script named `install` inside the binary tarball:

```
$ cd /tmp  
$ tar xfj nix-1.8-x86_64-darwin.tar.bz2  
$ cd nix-1.8-x86_64-darwin  
$ ./install
```

If you need to edit the multi-user installation script to use different group ID or a different user ID range, modify the variables set in the file named `install-multi-user`.

Installing Nix from Source

If no binary package is available or if you want to hack on Nix, you can build Nix from its Git repository.

Prerequisites

- GNU Autoconf (<https://www.gnu.org/software/autoconf/>) and the autoconf-archive macro collection (<https://www.gnu.org/software/autoconf-archive/>). These are needed to run the bootstrap script.
- GNU Make.
- Bash Shell. The `./configure` script relies on bashisms, so Bash is required.
- A version of GCC or Clang that supports C++20.
- `pkg-config` to locate dependencies. If your distribution does not provide it, you can get it from <http://www.freedesktop.org/wiki/Software/pkg-config>.
- The OpenSSL library to calculate cryptographic hashes. If your distribution does not provide it, you can get it from <https://www.openssl.org>.
- The `libbrotlienc` and `libbrotlidec` libraries to provide implementation of the Brotli compression algorithm. They are available for download from the official repository <https://github.com/google/brotli>.
- cURL and its library. If your distribution does not provide it, you can get it from <https://curl.haxx.se/>.
- The SQLite embedded database library, version 3.6.19 or higher. If your distribution does not provide it, please install it from <http://www.sqlite.org/>.
- The [Boehm garbage collector](#) to reduce the evaluator's memory consumption (optional). To enable it, install `pkgconfig` and the Boehm garbage collector, and pass the flag `--enable-gc` to `configure`.
- The `boost` library of version 1.66.0 or higher. It can be obtained from the official web site <https://www.boost.org/>.
- The `editline` library of version 1.14.0 or higher. It can be obtained from its repository <https://github.com/troglbit/editline>.
- The `libsodium` library for verifying cryptographic signatures of contents fetched from binary caches. It can be obtained from the official web site <https://libsodium.org>.
- Recent versions of Bison and Flex to build the parser. (This is because Nix needs GLR support in Bison and reentrancy support in Flex.) For Bison, you need version 2.6, which can be obtained from the [GNU FTP server](#). For Flex, you need version 2.5.35,

which is available on [SourceForge](#). Slightly older versions may also work, but ancient versions like the ubiquitous 2.5.4a won't.

- The `libseccomp` is used to provide syscall filtering on Linux. This is an optional dependency and can be disabled passing a `--disable-seccomp-sandboxing` option to the `configure` script (Not recommended unless your system doesn't support `libseccomp`). To get the library, visit <https://github.com/seccomp/libseccomp>.
- On 64-bit x86 machines only, `libcpuid` library is used to determine which microarchitecture levels are supported (e.g., as whether to have `x86_64-v2-linux` among additional system types). The library is available from its homepage <http://libcpuid.sourceforge.net>. This is an optional dependency and can be disabled by providing a `--disable-cpuid` to the `configure` script.
- Unless `./configure --disable-tests` is specified, GoogleTest (GTest) and RapidCheck are required, which are available at <https://google.github.io/googletest/> and <https://github.com/emil-e/rapidcheck> respectively.

Obtaining the Source

The most recent sources of Nix can be obtained from its [Git repository](#). For example, the following command will check out the latest revision into a directory called `nix`:

```
$ git clone https://github.com/NixOS/nix
```

Likewise, specific releases can be obtained from the [tags](#) of the repository.

Building Nix from Source

After cloning Nix's Git repository, issue the following commands:

```
$ ./bootstrap.sh  
$ ./configure options...  
$ make  
$ make install
```

Nix requires GNU Make so you may need to invoke `gmake` instead.

The installation path can be specified by passing the `--prefix=prefix` to `configure`. The default installation directory is `/usr/local`. You can change this to any location you like. You must have write permission to the `prefix` path.

Nix keeps its *store* (the place where packages are stored) in `/nix/store` by default. This can be changed using `--with-store-dir=path`.

Warning

It is best *not* to change the Nix store from its default, since doing so makes it impossible to use pre-built binaries from the standard Nixpkgs channels — that is, all packages will need to be built from source.

Nix keeps state (such as its database and log files) in `/nix/var` by default. This can be changed using `--localstatedir=path`.

Using Nix within Docker

To run the latest stable release of Nix with Docker run the following command:

```
$ docker run -ti nixos/nix
Unable to find image 'nixos/nix:latest' locally
latest: Pulling from nixos/nix
5843afab3874: Pull complete
b52bf13f109c: Pull complete
1e2415612aa3: Pull complete
Digest: sha256:27f6e7f60227e959ee7ece361f75d4844a40e1cc6878b6868fe30140420031ff
Status: Downloaded newer image for nixos/nix:latest
35ca4ada6e96:/# nix --version
nix (Nix) 2.3.12
35ca4ada6e96:/# exit
```

What is included in Nix's Docker image?

The official Docker image is created using `pkgs.dockerTools.buildLayeredImage` (and not with `Dockerfile` as it is usual with Docker images). You can still base your custom Docker image on it as you would do with any other Docker image.

The Docker image is also not based on any other image and includes minimal set of runtime dependencies that are required to use Nix:

- `pkgs.nix`
- `pkgs.bashInteractive`
- `pkgs.coreutils-full`
- `pkgs.gnutar`
- `pkgs.gzip`
- `pkgs.gnugrep`
- `pkgs.which`
- `pkgs.curl`
- `pkgs.less`
- `pkgs.wget`
- `pkgs.man`
- `pkgs.cacert.out`
- `pkgs.findutils`

Docker image with the latest development version of Nix

To get the latest image that was built by [Hydra](#) run the following command:

```
$ curl -L https://hydra.nixos.org/job/nix/master/dockerImage.x86_64-
linux/latest/download/1 | docker load
$ docker run -ti nix:2.5pre20211105
```

You can also build a Docker image from source yourself:

```
$ nix build ./\#hydraJobs.dockerImage.x86_64-linux
$ docker load -i ./result/image.tar.gz
$ docker run -ti nix:2.5pre20211105
```

Security

Nix has two basic security models. First, it can be used in “single-user mode”, which is similar to what most other package management tools do: there is a single user (typically root) who performs all package management operations. All other users can then use the installed packages, but they cannot perform package management operations themselves.

Alternatively, you can configure Nix in “multi-user mode”. In this model, all users can perform package management operations — for instance, every user can install software without requiring root privileges. Nix ensures that this is secure. For instance, it’s not possible for one user to overwrite a package used by another user with a Trojan horse.

Single-User Mode

In single-user mode, all Nix operations that access the database in `prefix/var/nix/db` or modify the Nix store in `prefix/store` must be performed under the user ID that owns those directories. This is typically root. (If you install from RPM packages, that's in fact the default ownership.) However, on single-user machines, it is often convenient to `chown` those directories to your normal user account so that you don't have to `su` to root all the time.

Multi-User Mode

To allow a Nix store to be shared safely among multiple users, it is important that users are not able to run builders that modify the Nix store or database in arbitrary ways, or that interfere with builds started by other users. If they could do so, they could install a Trojan horse in some package and compromise the accounts of other users.

To prevent this, the Nix store and database are owned by some privileged user (usually `root`) and builders are executed under special user accounts (usually named `nixbld1`, `nixbld2`, etc.). When a unprivileged user runs a Nix command, actions that operate on the Nix store (such as builds) are forwarded to a *Nix daemon* running under the owner of the Nix store/database that performs the operation.

Note

Multi-user mode has one important limitation: only root and a set of trusted users specified in `nix.conf` can specify arbitrary binary caches. So while unprivileged users may install packages from arbitrary Nix expressions, they may not get pre-built binaries.

Setting up the build users

The *build users* are the special UIDs under which builds are performed. They should all be members of the *build users group* `nixbld`. This group should have no other members. The build users should not be members of any other group. On Linux, you can create the group and users as follows:

```
$ groupadd -r nixbld
$ for n in $(seq 1 10); do useradd -c "Nix build user $n" \
    -d /var/empty -g nixbld -G nixbld -M -N -r -s "$(which nologin)" \
    nixbld$n; done
```

This creates 10 build users. There can never be more concurrent builds than the number of build users, so you may want to increase this if you expect to do many builds at the same time.

Running the daemon

The *Nix daemon* should be started as follows (as `root`):

```
$ nix-daemon
```

You'll want to put that line somewhere in your system's boot scripts.

To let unprivileged users use the daemon, they should set the `NIX_REMOTE` environment variable to `daemon`. So you should put a line like

```
export NIX_REMOTE=daemon
```

into the users' login scripts.

Restricting access

To limit which users can perform Nix operations, you can use the permissions on the directory `/nix/var/nix/daemon-socket`. For instance, if you want to restrict the use of Nix to the members of a group called `nix-users`, do

```
$ chgrp nix-users /nix/var/nix/daemon-socket
$ chmod ug=rwx,o= /nix/var/nix/daemon-socket
```

This way, users who are not in the `nix-users` group cannot connect to the Unix domain socket `/nix/var/nix/daemon-socket/socket`, so they cannot perform Nix operations.

Environment Variables

To use Nix, some environment variables should be set. In particular, `PATH` should contain the directories `prefix/bin` and `~/.nix-profile/bin`. The first directory contains the Nix tools themselves, while `~/.nix-profile` is a symbolic link to the current *user environment* (an automatically generated package consisting of symlinks to installed packages). The simplest way to set the required environment variables is to include the file `prefix/etc/profile.d/nix.sh` in your `~/.profile` (or similar), like this:

```
source prefix/etc/profile.d/nix.sh
```

NIX_SSL_CERT_FILE

If you need to specify a custom certificate bundle to account for an HTTPS-intercepting man in the middle proxy, you must specify the path to the certificate bundle in the environment variable `NIX_SSL_CERT_FILE`.

If you don't specify a `NIX_SSL_CERT_FILE` manually, Nix will install and use its own certificate bundle.

Set the environment variable and install Nix

```
$ export NIX_SSL_CERT_FILE=/etc/ssl/my-certificate-bundle.crt  
$ curl -L https://nixos.org/nix/install | sh
```

In the shell profile and rc files (for example, `/etc/bashrc`, `/etc/zshrc`), add the following line:

```
export NIX_SSL_CERT_FILE=/etc/ssl/my-certificate-bundle.crt
```

Note

You must not add the `export` and then do the `install`, as the Nix installer will detect the presence of Nix configuration, and abort.

If you use the Nix daemon, you should also add the following to `/etc/nix/nix.conf`:

```
ssl-cert-file = /etc/ssl/my-certificate-bundle.crt
```

Proxy Environment Variables

The Nix installer has special handling for these proxy-related environment variables:

`http_proxy`, `https_proxy`, `ftp_proxy`, `no_proxy`, `HTTP_PROXY`, `HTTPS_PROXY`, `FTP_PROXY`, `NO_PROXY`.

If any of these variables are set when running the Nix installer, then the installer will create an override file at `/etc/systemd/system/nix-daemon.service.d/override.conf` so `nix-daemon` will use them.

Upgrading Nix

Multi-user Nix users on macOS can upgrade Nix by running: `sudo -i sh -c 'nix-channel --update && nix-env --install --attr nixpkgs.nix && launchctl remove org.nixos.nix-daemon && launchctl load /Library/LaunchDaemons/org.nixos.nix-daemon.plist'`

Single-user installations of Nix should run this: `nix-channel --update; nix-env --install --attr nixpkgs.nix nixpkgs.cacert`

Multi-user Nix users on Linux should run this with sudo: `nix-channel --update; nix-env --install --attr nixpkgs.nix nixpkgs.cacert; systemctl daemon-reload; systemctl restart nix-daemon`

Uninstalling Nix

Single User

If you have a [single-user installation](#) of Nix, uninstall it by running:

```
$ rm -rf /nix
```

Multi User

Removing a [multi-user installation](#) of Nix is more involved, and depends on the operating system.

Linux

If you are on Linux with systemd:

1. Remove the Nix daemon service:

```
sudo systemctl stop nix-daemon.service  
sudo systemctl disable nix-daemon.socket nix-daemon.service  
sudo systemctl daemon-reload
```

Remove files created by Nix:

```
sudo rm -rf /etc/nix /etc/profile.d/nix.sh /etc/tmpfiles.d/nix-daemon.conf /nix  
~root/.nix-channels ~root/.nix-defexpr ~root/.nix-profile
```

Remove build users and their group:

```
for i in $(seq 1 32); do  
    sudo userdel nixbld$i  
done  
sudo groupdel nixbld
```

There may also be references to Nix in

- [/etc/bash.bashrc](#)
- [/etc/bashrc](#)

- `/etc/profile`
- `/etc/zsh/zshrc`
- `/etc/zshrc`

which you may remove.

macOS

1. Edit `/etc/zshrc`, `/etc/bashrc`, and `/etc/bash.bashrc` to remove the lines sourcing `nix-daemon.sh`, which should look like this:

```
# Nix
if [ -e '/nix/var/nix/profiles/default/etc/profile.d/nix-daemon.sh' ]; then
    . '/nix/var/nix/profiles/default/etc/profile.d/nix-daemon.sh'
fi
# End Nix
```

If these files haven't been altered since installing Nix you can simply put the backups back in place:

```
sudo mv /etc/zshrc.backup-before-nix /etc/zshrc
sudo mv /etc/bashrc.backup-before-nix /etc/bashrc
sudo mv /etc/bash.bashrc.backup-before-nix /etc/bash.bashrc
```

This will stop shells from sourcing the file and bringing everything you installed using Nix in scope.

2. Stop and remove the Nix daemon services:

```
sudo launchctl unload /Library/LaunchDaemons/org.nixos.nix-daemon.plist
sudo rm /Library/LaunchDaemons/org.nixos.nix-daemon.plist
sudo launchctl unload /Library/LaunchDaemons/org.nixos.darwin-store.plist
sudo rm /Library/LaunchDaemons/org.nixos.darwin-store.plist
```

This stops the Nix daemon and prevents it from being started next time you boot the system.

3. Remove the `nixbld` group and the `_nixbuildN` users:

```
sudo dscl . -delete /Groups/nixbld  
for u in $(sudo dscl . -list /Users | grep _nixbld); do sudo dscl . -delete  
/Users/$u; done
```

This will remove all the build users that no longer serve a purpose.

4. Edit fstab using `sudo vi /etc/fstab` to remove the line mounting the Nix Store volume on `/nix`, which looks like `UUID=<uuid> /nix apfs rw,noauto,nobrowse,suid,owners` or `LABEL=Nix\040Store /nix apfs rw,nobrowse`. This will prevent automatic mounting of the Nix Store volume.
5. Edit `/etc/synthetic.conf` to remove the `nix` line. If this is the only line in the file you can remove it entirely, `sudo rm /etc/synthetic.conf`. This will prevent the creation of the empty `/nix` directory to provide a mountpoint for the Nix Store volume.
6. Remove the files Nix added to your system:

```
sudo rm -rf /etc/nix /var/root/.nix-profile /var/root/.nix-defexpr  
/var/root/.nix-channels ~/.nix-profile ~/.nix-defexpr ~/.nix-channels
```

This gets rid of any data Nix may have created except for the store which is removed next.

7. Remove the Nix Store volume:

```
sudo diskutil apfs deleteVolume /nix
```

This will remove the Nix Store volume and everything that was added to the store.

If the output indicates that the command couldn't remove the volume, you should make sure you don't have an *unmounted* Nix Store volume. Look for a "Nix Store" volume in the output of the following command:

```
diskutil list
```

If you *do* see a "Nix Store" volume, delete it by re-running the `diskutil deleteVolume` command, but replace `/nix` with the store volume's `diskXsY` identifier.

Note

After you complete the steps here, you will still have an empty `/nix` directory. This is

an expected sign of a successful uninstall. The empty `/nix` directory will disappear the next time you reboot.

You do not have to reboot to finish uninstalling Nix. The uninstall is complete. macOS (Catalina+) directly controls root directories and its read-only root will prevent you from manually deleting the empty `/nix` mountpoint.

This chapter discusses how to do package management with Nix, i.e., how to obtain, install, upgrade, and erase packages. This is the “user’s” perspective of the Nix system — people who want to *create* packages should consult the chapter on the [Nix language](#).

Basic Package Management

The main command for package management is `nix-env`. You can use it to install, upgrade, and erase packages, and to query what packages are installed or are available for installation.

In Nix, different users can have different “views” on the set of installed applications. That is, there might be lots of applications present on the system (possibly in many different versions), but users can have a specific selection of those active — where “active” just means that it appears in a directory in the user’s `PATH`. Such a view on the set of installed applications is called a *user environment*, which is just a directory tree consisting of symlinks to the files of the active applications.

Components are installed from a set of *Nix expressions* that tell Nix how to build those packages, including, if necessary, their dependencies. There is a collection of Nix expressions called the Nixpkgs package collection that contains packages ranging from basic development stuff such as GCC and Glibc, to end-user applications like Mozilla Firefox. (Nix is however not tied to the Nixpkgs package collection; you could write your own Nix expressions based on Nixpkgs, or completely new ones.)

You can manually download the latest version of Nixpkgs from <https://github.com/NixOS/nixpkgs>. However, it’s much more convenient to use the Nixpkgs *channel*, since it makes it easy to stay up to date with new versions of Nixpkgs. Nixpkgs is automatically added to your list of “subscribed” channels when you install Nix. If this is not the case for some reason, you can add it as follows:

```
$ nix-channel --add https://nixos.org/channels/nixpkgs-unstable  
$ nix-channel --update
```

Note

On NixOS, you’re automatically subscribed to a NixOS channel corresponding to your NixOS major release (e.g. <http://nixos.org/channels/nixos-21.11>). A NixOS channel is identical to the Nixpkgs channel, except that it contains only Linux binaries and is updated only if a set of regression tests succeed.

You can view the set of available packages in Nixpkgs:

```
$ nix-env --query --available --attr-path
nixpkgs.aterm                      aterm-2.2
nixpkgs.bash                        bash-3.0
nixpkgs.binutils                     binutils-2.15
nixpkgs.bison                        bison-1.875d
nixpkgs.blackdown                   blackdown-1.4.2
nixpkgs.bzip2                       bzip2-1.0.2
...
...
```

The flag `-q` specifies a query operation, `-a` means that you want to show the “available” (i.e., installable) packages, as opposed to the installed packages, and `-P` prints the attribute paths that can be used to unambiguously select a package for installation (listed in the first column). If you downloaded Nixpkgs yourself, or if you checked it out from GitHub, then you need to pass the path to your Nixpkgs tree using the `-f` flag:

```
$ nix-env --query --available --attr-path --file /path/to/nixpkgs
aterm                           aterm-2.2
bash                            bash-3.0
...
...
```

where `/path/to/nixpkgs` is where you’ve unpacked or checked out Nixpkgs.

You can filter the packages by name:

```
$ nix-env --query --available --attr-path firefox
nixpkgs.firefox-esr                firefox-91.3.0esr
nixpkgs.firefox                    firefox-94.0.1
```

and using regular expressions:

```
$ nix-env --query --available --attr-path 'firefox.*'
```

It is also possible to see the *status* of available packages, i.e., whether they are installed into the user environment and/or present in the system:

```
$ nix-env --query --available --attr-path --status
...
-PS  nixpkgs.bash                  bash-3.0
--S  nixpkgs.binutils              binutils-2.15
IPS  nixpkgs.bison                bison-1.875d
...
...
```

The first character (`I`) indicates whether the package is installed in your current user environment. The second (`P`) indicates whether it is present on your system (in which case

installing it into your user environment would be a very quick operation). The last one (`S`) indicates whether there is a so-called *substitute* for the package, which is Nix’s mechanism for doing binary deployment. It just means that Nix knows that it can fetch a pre-built package from somewhere (typically a network server) instead of building it locally.

You can install a package using `nix-env --install --attr .`. For instance,

```
$ nix-env --install --attr nixpkgs.subversion
```

will install the package called `subversion` from `nixpkgs` channel (which is, of course, the [Subversion version management system](#)).

Note

When you ask Nix to install a package, it will first try to get it in pre-compiled form from a *binary cache*. By default, Nix will use the binary cache <https://cache.nixos.org>; it contains binaries for most packages in Nixpkgs. Only if no binary is available in the binary cache, Nix will build the package from source. So if `nix-env -iA nixpkgs.subversion` results in Nix building stuff from source, then either the package is not built for your platform by the Nixpkgs build servers, or your version of Nixpkgs is too old or too new. For instance, if you have a very recent checkout of Nixpkgs, then the Nixpkgs build servers may not have had a chance to build everything and upload the resulting binaries to <https://cache.nixos.org>. The Nixpkgs channel is only updated after all binaries have been uploaded to the cache, so if you stick to the Nixpkgs channel (rather than using a Git checkout of the Nixpkgs tree), you will get binaries for most packages.

Naturally, packages can also be uninstalled. Unlike when installing, you will need to use the derivation name (though the version part can be omitted), instead of the attribute path, as `nix-env` does not record which attribute was used for installing:

```
$ nix-env --uninstall subversion
```

Upgrading to a new version is just as easy. If you have a new release of Nix Packages, you can do:

```
$ nix-env --upgrade --attr nixpkgs.subversion
```

This will *only* upgrade Subversion if there is a “newer” version in the new set of Nix expressions, as defined by some pretty arbitrary rules regarding ordering of version

numbers (which generally do what you'd expect of them). To just unconditionally replace Subversion with whatever version is in the Nix expressions, use `-i` instead of `-u`; `-i` will remove whatever version is already installed.

You can also upgrade all packages for which there are newer versions:

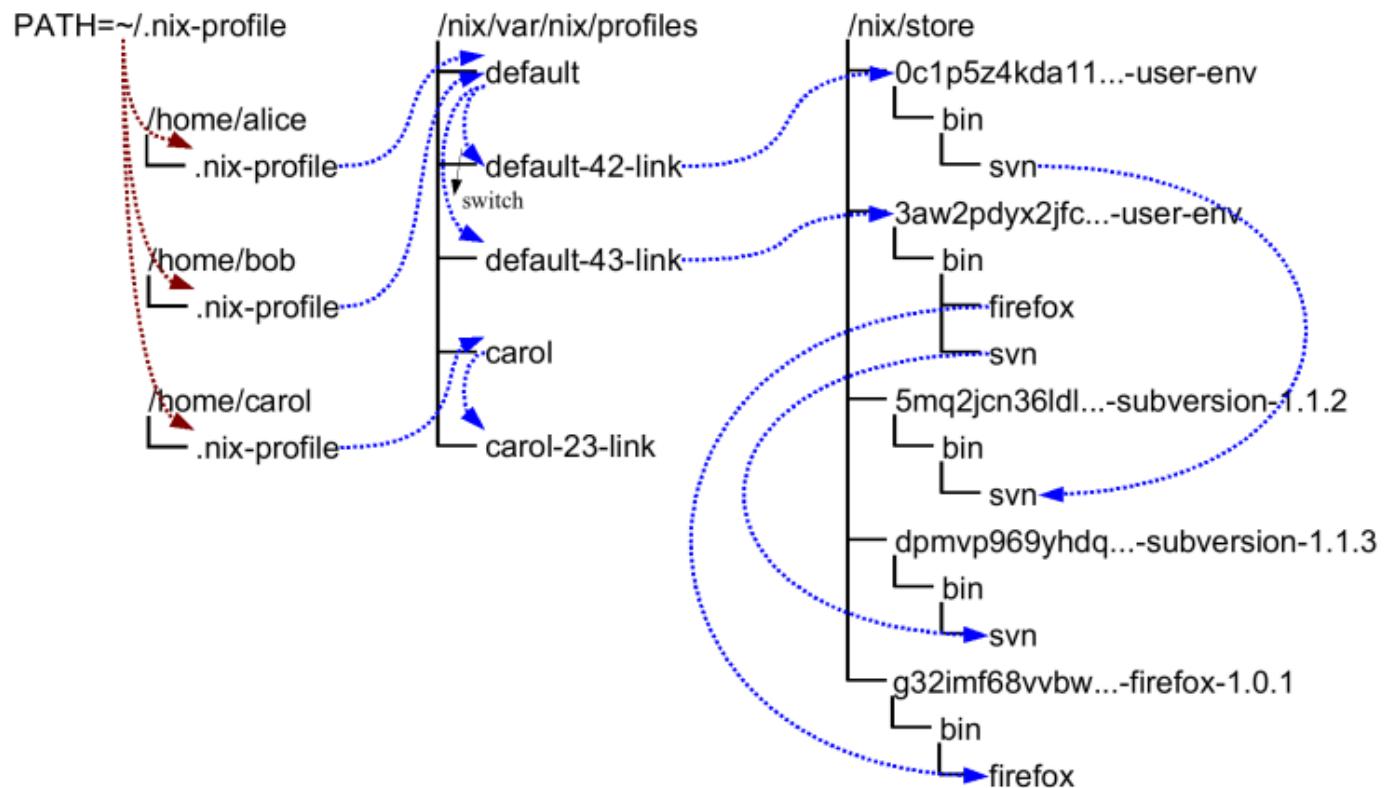
```
$ nix-env --upgrade
```

Sometimes it's useful to be able to ask what `nix-env` would do, without actually doing it. For instance, to find out what packages would be upgraded by `nix-env --upgrade`, you can do

```
$ nix-env --upgrade --dry-run
(dry run; not doing anything)
upgrading `libxslt-1.1.0' to `libxslt-1.1.10'
upgrading `graphviz-1.10' to `graphviz-1.12'
upgrading `coreutils-5.0' to `coreutils-5.2.1'
```

Profiles

Profiles and user environments are Nix's mechanism for implementing the ability to allow different users to have different configurations, and to do atomic upgrades and rollbacks. To understand how they work, it's useful to know a bit about how Nix works. In Nix, packages are stored in unique locations in the *Nix store* (typically, `/nix/store`). For instance, a particular version of the Subversion package might be stored in a directory `/nix/store/dpmvp969yhdq...-subversion-1.1.3/`, while another version might be stored in `/nix/store/5mq2jcn36ldlmh93yj1n8s9c95pj7c5s-subversion-1.1.2`. The long strings prefixed to the directory names are cryptographic hashes (to be precise, 160-bit truncations of SHA-256 hashes encoded in a base-32 notation) of *all* inputs involved in building the package — sources, dependencies, compiler flags, and so on. So if two packages differ in any way, they end up in different locations in the file system, so they don't interfere with each other. Here is what a part of a typical Nix store looks like:



Of course, you wouldn't want to type

```
$ /nix/store/dpmvp969yhdq...-subversion-1.1.3/bin/svn
```

every time you want to run Subversion. Of course we could set up the `PATH` environment variable to include the `bin` directory of every package we want to use, but this is not very convenient since changing `PATH` doesn't take effect for already existing processes. The

solution Nix uses is to create directory trees of symlinks to *activated* packages. These are called *user environments* and they are packages themselves (though automatically generated by `nix-env`), so they too reside in the Nix store. For instance, in the figure above, the user environment `/nix/store/0c1p5z4kda11...-user-env` contains a symlink to just Subversion 1.1.2 (arrows in the figure indicate symlinks). This would be what we would obtain if we had done

```
$ nix-env --install --attr nixpkgs.subversion
```

on a set of Nix expressions that contained Subversion 1.1.2.

This doesn't in itself solve the problem, of course; you wouldn't want to type `/nix/store/0c1p5z4kda11...-user-env/bin svn` either. That's why there are symlinks outside of the store that point to the user environments in the store; for instance, the symlinks `default-42-link` and `default-43-link` in the example. These are called *generations* since every time you perform a `nix-env` operation, a new user environment is generated based on the current one. For instance, generation 43 was created from generation 42 when we did

```
$ nix-env --install --attr nixpkgs.subversion nixpkgs.firefox
```

on a set of Nix expressions that contained Firefox and a new version of Subversion.

Generations are grouped together into *profiles* so that different users don't interfere with each other if they don't want to. For example:

```
$ ls -l /nix/var/nix/profiles/
...
lrwxrwxrwx 1 eelco ... default-42-link -> /nix/store/0c1p5z4kda11...-user-env
lrwxrwxrwx 1 eelco ... default-43-link -> /nix/store/3aw2pdyx2jfc...-user-env
lrwxrwxrwx 1 eelco ... default -> default-43-link
```

This shows a profile called `default`. The file `default` itself is actually a symlink that points to the current generation. When we do a `nix-env` operation, a new user environment and generation link are created based on the current one, and finally the `default` symlink is made to point at the new generation. This last step is atomic on Unix, which explains how we can do atomic upgrades. (Note that the building/installing of new packages doesn't interfere in any way with old packages, since they are stored in different locations in the Nix store.)

If you find that you want to undo a `nix-env` operation, you can just do

```
$ nix-env --rollback
```

which will just make the current generation link point at the previous link. E.g., `default` would be made to point at `default-42-link`. You can also switch to a specific generation:

```
$ nix-env --switch-generation 43
```

which in this example would roll forward to generation 43 again. You can also see all available generations:

```
$ nix-env --list-generations
```

You generally wouldn't have `/nix/var/nix/profiles/some-profile/bin` in your `PATH`. Rather, there is a symlink `~/.nix-profile` that points to your current profile. This means that you should put `~/.nix-profile/bin` in your `PATH` (and indeed, that's what the initialisation script `/nix/etc/profile.d/nix.sh` does). This makes it easier to switch to a different profile. You can do that using the command `nix-env --switch-profile`:

```
$ nix-env --switch-profile /nix/var/nix/profiles/my-profile
```

```
$ nix-env --switch-profile /nix/var/nix/profiles/default
```

These commands switch to the `my-profile` and default profile, respectively. If the profile doesn't exist, it will be created automatically. You should be careful about storing a profile in another location than the `profiles` directory, since otherwise it might not be used as a root of the [garbage collector](#).

All `nix-env` operations work on the profile pointed to by `~/.nix-profile`, but you can override this using the `--profile` option (abbreviation `-p`):

```
$ nix-env --profile /nix/var/nix/profiles/other-profile --install --attr  
nixpkgs.subversion
```

This will *not* change the `~/.nix-profile` symlink.

Garbage Collection

`nix-env` operations such as upgrades (`-u`) and uninstall (`-e`) never actually delete packages from the system. All they do (as shown above) is to create a new user environment that no longer contains symlinks to the “deleted” packages.

Of course, since disk space is not infinite, unused packages should be removed at some point. You can do this by running the Nix garbage collector. It will remove from the Nix store any package not used (directly or indirectly) by any generation of any profile.

Note however that as long as old generations reference a package, it will not be deleted. After all, we wouldn’t be able to do a rollback otherwise. So in order for garbage collection to be effective, you should also delete (some) old generations. Of course, this should only be done if you are certain that you will not need to roll back.

To delete all old (non-current) generations of your current profile:

```
$ nix-env --delete-generations old
```

Instead of `old` you can also specify a list of generations, e.g.,

```
$ nix-env --delete-generations 10 11 14
```

To delete all generations older than a specified number of days (except the current generation), use the `d` suffix. For example,

```
$ nix-env --delete-generations 14d
```

deletes all generations older than two weeks.

After removing appropriate old generations you can run the garbage collector as follows:

```
$ nix-store --gc
```

The behaviour of the garbage collector is affected by the `keep-derivations` (default: true) and `keep-outputs` (default: false) options in the Nix configuration file. The defaults will ensure that all derivations that are build-time dependencies of garbage collector roots will be kept and that all output paths that are runtime dependencies will be kept as well. All other derivations or paths will be collected. (This is usually what you want, but while you are developing it may make sense to keep outputs to ensure that rebuild times are quick.) If you are feeling uncertain, you can also first view what files would be deleted:

```
$ nix-store --gc --print-dead
```

Likewise, the option `--print-live` will show the paths that *won't* be deleted.

There is also a convenient little utility `nix-collect-garbage`, which when invoked with the `-d` (`--delete-old`) switch deletes all old generations of all profiles in `/nix/var/nix/profiles`. So

```
$ nix-collect-garbage -d
```

is a quick and easy way to clean up your system.

Garbage Collector Roots

The roots of the garbage collector are all store paths to which there are symlinks in the directory `prefix/nix/var/nix/gcroots`. For instance, the following command makes the path `/nix/store/d718ef...-foo` a root of the collector:

```
$ ln -s /nix/store/d718ef...-foo /nix/var/nix/gcroots/bar
```

That is, after this command, the garbage collector will not remove `/nix/store/d718ef...-foo` or any of its dependencies.

Subdirectories of `prefix/nix/var/nix/gcroots` are also searched for symlinks. Symlinks to non-store paths are followed and searched for roots, but symlinks to non-store paths *inside* the paths reached in that way are not followed to prevent infinite recursion.

Sharing Packages Between Machines

Sometimes you want to copy a package from one machine to another. Or, you want to install some packages and you know that another machine already has some or all of those packages or their dependencies. In that case there are mechanisms to quickly copy packages between machines.

Serving a Nix store via HTTP

You can easily share the Nix store of a machine via HTTP. This allows other machines to fetch store paths from that machine to speed up installations. It uses the same *binary cache* mechanism that Nix usually uses to fetch pre-built binaries from <https://cache.nixos.org>.

The daemon that handles binary cache requests via HTTP, `nix-serve`, is not part of the Nix distribution, but you can install it from Nixpkgs:

```
$ nix-env --install --attr nixpkgs.nix-serve
```

You can then start the server, listening for HTTP connections on whatever port you like:

```
$ nix-serve -p 8080
```

To check whether it works, try the following on the client:

```
$ curl http://avalon:8080/nix-cache-info
```

which should print something like:

```
StoreDir: /nix/store
WantMassQuery: 1
Priority: 30
```

On the client side, you can tell Nix to use your binary cache using `--substituters`, e.g.:

```
$ nix-env --install --attr nixpkgs.firefox --substituters http://avalon:8080/
```

The option `substituters` tells Nix to use this binary cache in addition to your default caches, such as <https://cache.nixos.org>. Thus, for any path in the closure of Firefox, Nix will first check if the path is available on the server `avalon` or another binary caches. If not, it will fall back to building from source.

You can also tell Nix to always use your binary cache by adding a line to the `nix.conf` configuration file like this:

```
substituters = http://avalon:8080/ https://cache.nixos.org/
```

Copying Closures via SSH

The command `nix-copy-closure` copies a Nix store path along with all its dependencies to or from another machine via the SSH protocol. It doesn't copy store paths that are already present on the target machine. For example, the following command copies Firefox with all its dependencies:

```
$ nix-copy-closure --to alice@itchy.example.org $(type -p firefox)
```

See the [manpage](#) for `nix-copy-closure` for details.

With `nix-store --export` and `nix-store --import` you can write the closure of a store path (that is, the path and all its dependencies) to a file, and then unpack that file into another Nix store. For example,

```
$ nix-store --export $(nix-store --query --requisites $(type -p firefox)) > firefox.closure
```

writes the closure of Firefox to a file. You can then copy this file to another machine and install the closure:

```
$ nix-store --import < firefox.closure
```

Any store paths in the closure that are already present in the target store are ignored. It is also possible to pipe the export into another command, e.g. to copy and install a closure directly to/on another machine:

```
$ nix-store --export $(nix-store --query --requisites $(type -p firefox)) | bzip2 | \
    ssh alice@itchy.example.org "bunzip2 | nix-store --import"
```

However, `nix-copy-closure` is generally more efficient because it only copies paths that are not already present in the target Nix store.

Serving a Nix store via SSH

You can tell Nix to automatically fetch needed binaries from a remote Nix store via SSH. For example, the following installs Firefox, automatically fetching any store paths in Firefox's closure if they are available on the server `avalon`:

```
$ nix-env --install --attr nixpkgs.firefox --substituters ssh://alice@avalon
```

This works similar to the binary cache substituter that Nix usually uses, only using SSH instead of HTTP: if a store path `P` is needed, Nix will first check if it's available in the Nix store on `avalon`. If not, it will fall back to using the binary cache substituter, and then to building from source.

Note

The SSH substituter currently does not allow you to enter an SSH passphrase interactively. Therefore, you should use `ssh-add` to load the decrypted private key into `ssh-agent`.

You can also copy the closure of some store path, without installing it into your profile, e.g.

```
$ nix-store --realise /nix/store/m85bxg...-firefox-34.0.5 --substituters  
ssh://alice@avalon
```

This is essentially equivalent to doing

```
$ nix-copy-closure --from alice@avalon  
/nix/store/m85bxg...-firefox-34.0.5
```

You can use SSH's *forced command* feature to set up a restricted user account for SSH substituter access, allowing read-only access to the local Nix store, but nothing more. For example, add the following lines to `sshd_config` to restrict the user `nix-ssh`:

```
Match User nix-ssh  
  AllowAgentForwarding no  
  AllowTcpForwarding no  
  PermitTTY no  
  PermitTunnel no  
  X11Forwarding no  
  ForceCommand nix-store --serve  
Match All
```

On NixOS, you can accomplish the same by adding the following to your `configuration.nix`:

```
nix.sshServe.enable = true;  
nix.sshServe.keys = [ "ssh-dss AAAAB3NzaC1k... bob@example.org" ];
```

where the latter line lists the public keys of users that are allowed to connect.

Serving a Nix store via S3

Nix has [built-in support](#) for storing and fetching store paths from Amazon S3 and S3-compatible services. This uses the same *binary* cache mechanism that Nix usually uses to fetch prebuilt binaries from cache.nixos.org.

In this example we will use the bucket named `example-nix-cache`.

Anonymous Reads to your S3-compatible binary cache

If your binary cache is publicly accessible and does not require authentication, the simplest and easiest way to use Nix with your S3 compatible binary cache is to use the HTTP URL for that cache.

For AWS S3 the binary cache URL for example bucket will be exactly <https://example-nix-cache.s3.amazonaws.com> or `s3://example-nix-cache`. For S3 compatible binary caches, consult that cache's documentation.

Your bucket will need the following bucket policy:

```
{
  "Id": "DirectReads",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDirectReads",
      "Action": [
        "s3:GetObject",
        "s3:GetBucketLocation"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::example-nix-cache",
        "arn:aws:s3:::example-nix-cache/*"
      ],
      "Principal": "*"
    }
  ]
}
```

Authenticated Reads to your S3 binary cache

For AWS S3 the binary cache URL for example bucket will be exactly `s3://example-nix-cache`.

Nix will use the [default credential provider chain](#) for authenticating requests to Amazon S3.

Nix supports authenticated reads from Amazon S3 and S3 compatible binary caches.

Your bucket will need a bucket policy allowing the desired users to perform the `s3:GetObject` and `s3:GetBucketLocation` action on all objects in the bucket. The [anonymous policy given above](#) can be updated to have a restricted `Principal` to support this.

Authenticated Writes to your S3-compatible binary cache

Nix support fully supports writing to Amazon S3 and S3 compatible buckets. The binary cache URL for our example bucket will be <s3://example-nix-cache>.

Nix will use the [default credential provider chain](#) for authenticating requests to Amazon S3.

Your account will need the following IAM policy to upload to the cache:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UploadToCache",
      "Effect": "Allow",
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:GetBucketLocation",
        "s3:GetObject",
        "s3>ListBucket",
        "s3>ListBucketMultipartUploads",
        "s3>ListMultipartUploadParts",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::example-nix-cache",
        "arn:aws:s3:::example-nix-cache/*"
      ]
    }
  ]
}
```

Examples

To upload with a specific credential profile for Amazon S3:

```
$ nix copy nixpkgs.hello \
--to 's3://example-nix-cache?profile=cache-upload&region=eu-west-2'
```

To upload to an S3-compatible binary cache:

```
$ nix copy nixpkgs.hello --to \
's3://example-nix-cache?profile=cache-upload&scheme=https&
endpoint=minio.example.com'
```

Nix Language

The Nix language is designed for conveniently creating and composing *derivations* – precise descriptions of how contents of existing files are used to derive new files. It is:

- *domain-specific*

It comes with [built-in functions](#) to integrate with the Nix store, which manages files and performs the derivations declared in the Nix language.

- *declarative*

There is no notion of executing sequential steps. Dependencies between operations are established only through data.

- *pure*

Values cannot change during computation. Functions always produce the same output if their input does not change.

- *functional*

Functions are like any other value. Functions can be assigned to names, taken as arguments, or returned by functions.

- *lazy*

Values are only computed when they are needed.

- *dynamically typed*

Type errors are only detected when expressions are evaluated.

Overview

This is an incomplete overview of language features, by example.

Example	Description
<i>Basic values</i>	

"hello world"	A string
'' multi line string ''	A multi-line string. Strips common prefixed whitespace. Evaluates to "multi\n line\n string".
"hello \${ { a = "world"; } .a }" "1 2 \${toString 3}" "\${pkgs.bash}/bin/sh"	String interpolation (expands to "hello world", "1 2 3", "/nix/store/<hash>-bash-<version>/bin/sh")
true, false	Booleans
null	Null value
123	An integer
3.141	A floating point number
/etc	An absolute path
./foo.png	A path relative to the file containing this Nix expression
~/.config	A home path. Evaluates to the "<user's home directory>/.config".
<nixpkgs>	Search path for Nix files. Value determined by \$NIX_PATH environment variable .

<i>Compound values</i>	
{ x = 1; y = 2; }	A set with attributes named <code>x</code> and <code>y</code>
{ foo.bar = 1; }	A nested set, equivalent to { foo = { bar = 1; }; }
rec { x = "foo"; y = x + "bar"; }	A recursive set, equivalent to { x = "foo"; y = "foobar"; }
["foo" "bar" "baz"] [1 2 3] [(f 1) { a = 1; b = 2; } ["c"]]	Lists with three elements.
<i>Operators</i>	
"foo" + "bar"	String concatenation
1 + 2	Integer addition
"foo" == "f" + "oo"	Equality test (evaluates to <code>true</code>)
"foo" != "bar"	Inequality test (evaluates to <code>true</code>)
!true	Boolean negation
{ x = 1; y = 2; }.x	Attribute selection (evaluates to <code>1</code>)

<code>{ x = 1; y = 2; }.z or 3</code>	Attribute selection with default (evaluates to <code>3</code>)
<code>{ x = 1; y = 2; } // { z = 3; }</code>	Merge two sets (attributes in the right-hand set taking precedence)
<i>Control structures</i>	
<code>if 1 + 1 == 2 then "yes!" else "no!"</code>	Conditional expression
<code>assert 1 + 1 == 2; "yes!"</code>	Assertion check (evaluates to <code>"yes!"</code>).
<code>let x = "foo"; y = "bar"; in x + y</code>	Variable definition
<code>with builtins; head [1 2 3]</code>	Add all attributes from the given set to the scope (evaluates to <code>1</code>)
<i>Functions (lambdas)</i>	
<code>x: x + 1</code>	A function that expects an integer and returns it increased by 1
<code>x: y: x + y</code>	Curried function, equivalent to <code>x: (y: x + y)</code> . Can be used like a function that takes two arguments and returns their sum.
<code>(x: x + 1) 100</code>	A function call (evaluates to 101)
<code>let inc = x: x + 1; in inc (inc (inc 100))</code>	A function bound to a variable and subsequently called by name (evaluates to 103)

{ x, y }: x + y	A function that expects a set with required attributes <code>x</code> and <code>y</code> and concatenates them
{ x, y ? "bar" }: x + y	A function that expects a set with required attribute <code>x</code> and optional <code>y</code> , using <code>"bar"</code> as default value for <code>y</code>
{ x, y, ... }: x + y	A function that expects a set with required attributes <code>x</code> and <code>y</code> and ignores any other attributes
{ x, y } @ args: x + y args @ { x, y }: x + y	A function that expects a set with required attributes <code>x</code> and <code>y</code> , and binds the whole set to <code>args</code>
<i>Built-in functions</i>	
import ./foo.nix	Load and return Nix expression in given file
map (x: x + x) [1 2 3]	Apply a function to every element of a list (evaluates to [2 4 6])

Data Types

Primitives

- [String](#)

Strings can be written in three ways.

The most common way is to enclose the string between double quotes, e.g., `"foo
bar"`. Strings can span multiple lines. The special characters `"` and `\` and the character sequence `${}` must be escaped by prefixing them with a backslash (`\`). Newlines, carriage returns and tabs can be written as `\n`, `\r` and `\t`, respectively.

You can include the results of other expressions into a string by enclosing them in `${}`, a feature known as [string interpolation](#).

The second way to write string literals is as an *indented string*, which is enclosed between pairs of *double single-quotes*, like so:

```
''  
  This is the first line.  
  This is the second line.  
  This is the third line.  
'''
```

This kind of string literal intelligently strips indentation from the start of each line. To be precise, it strips from each line a number of spaces equal to the minimal indentation of the string as a whole (disregarding the indentation of empty lines). For instance, the first and second line are indented two spaces, while the third line is indented four spaces. Thus, two spaces are stripped from each line, so the resulting string is

```
"This is the first line.\nThis is the second line.\n  This is the third  
line.\n"
```

Note that the whitespace and newline following the opening `''` is ignored if there is no non-whitespace text on the initial line.

Indented strings support [string interpolation](#).

Since `${}` and `''` have special meaning in indented strings, you need a way to quote

them. `$` can be escaped by prefixing it with `''` (that is, two single quotes), i.e., `''$`. `''` can be escaped by prefixing it with `'`, i.e., `'''`. `$` removes any special meaning from the following `$`. Linefeed, carriage-return and tab characters can be written as `\n`, `\r`, `\t`, and `\` escapes any other character.

Indented strings are primarily useful in that they allow multi-line string literals to follow the indentation of the enclosing Nix expression, and that less escaping is typically necessary for strings representing languages such as shell scripts and configuration files because `''` is much less common than `"`. Example:

```
stdenv.mkDerivation {  
  ...  
  postInstall =  
  ''  
    mkdir $out/bin $out/etc  
    cp foo $out/bin  
    echo "Hello World" > $out/etc/foo.conf  
    ${if enableBar then "cp bar $out/bin" else ""}  
  '';  
  ...  
}
```

Finally, as a convenience, *URLs* as defined in appendix B of [RFC 2396](#) can be written *as is*, without quotes. For instance, the string `"http://example.org/foo.tar.bz2"` can also be written as `http://example.org/foo.tar.bz2`.

- [Number](#)

Numbers, which can be *integers* (like `123`) or *floating point* (like `123.43` or `.27e13`).

See [arithmetic](#) and [comparison operators](#) for semantics.

- [Path](#)

Paths, e.g., `/bin/sh` or `./builder.sh`. A path must contain at least one slash to be recognised as such. For instance, `builder.sh` is not a path: it's parsed as an expression that selects the attribute `sh` from the variable `builder`. If the file name is relative, i.e., if it does not begin with a slash, it is made absolute at parse time relative to the directory of the Nix expression that contained it. For instance, if a Nix expression in `/foo/bar/bla.nix` refers to `../xyzzy/fnord.nix`, the absolute path is `/foo/xyzzy/fnord.nix`.

If the first component of a path is a `~`, it is interpreted as if the rest of the path were relative to the user's home directory. e.g. `~/foo` would be equivalent to `/home/edolstra/foo` for a user whose home directory is `/home/edolstra`.

Paths can also be specified between angle brackets, e.g. `<nixpkgs>`. This means that the directories listed in the environment variable `NIX_PATH` will be searched for the given file or directory name.

When an [interpolated string](#) evaluates to a path, the path is first copied into the Nix store and the resulting string is the [store path](#) of the newly created [store object](#).

For instance, evaluating `"${./foo.txt}"` will cause `foo.txt` in the current directory to be copied into the Nix store and result in the string `"/nix/store/<hash>-foo.txt"`.

Note that the Nix language assumes that all input files will remain *unchanged* while evaluating a Nix expression. For example, assume you used a file path in an interpolated string during a `nix repl` session. Later in the same session, after having changed the file contents, evaluating the interpolated string with the file path again might not return a new store path, since Nix might not re-read the file contents.

Paths themselves, except those in angle brackets (`< >`), support [string interpolation](#).

At least one slash (`/`) must appear *before* any interpolated expression for the result to be recognized as a path.

`a.${foo}/b.${bar}` is a syntactically valid division operation. `./a.${foo}/b.${bar}` is a path.

- [Boolean](#)

Booleans with values `true` and `false`.

- [Null](#)

The null value, denoted as `null`.

List

Lists are formed by enclosing a whitespace-separated list of values between square brackets. For example,

```
[ 123 ./foo.nix "abc" (f { x = y; }) ]
```

defines a list of four elements, the last being the result of a call to the function `f`. Note that function calls have to be enclosed in parentheses. If they had been omitted, e.g.,

```
[ 123 ./foo.nix "abc" f { x = y; } ]
```

the result would be a list of five elements, the fourth one being a function and the fifth being a set.

Note that lists are only lazy in values, and they are strict in length.

Attribute Set

An attribute set is a collection of name-value-pairs (called *attributes*) enclosed in curly brackets (`{ }`).

An attribute name can be an identifier or a [string](#). An identifier must start with a letter (`a-z`, `A-Z`) or underscore (`_`), and can otherwise contain letters (`a-z`, `A-Z`), numbers (`0-9`), underscores (`_`), apostrophes (`'`), or dashes (`-`).

```
name = identifier | string  
identifier ~ [a-zA-Z_][a-zA-Z0-9_-]*
```

Names and values are separated by an equal sign (`=`). Each value is an arbitrary expression terminated by a semicolon (`;`).

```
attrset = { [ name = expr ; ] ... }
```

Attributes can appear in any order. An attribute name may only occur once.

Example:

```
{  
  x = 123;  
  text = "Hello";  
  y = f { bla = 456; };  
}
```

This defines a set with attributes named `x`, `text`, `y`.

Attributes can be accessed with the [. operator](#).

Example:

```
{ a = "Foo"; b = "Bar"; }.a
```

This evaluates to `"Foo"`.

It is possible to provide a default value in an attribute selection using the `or` keyword.

Example:

```
{ a = "Foo"; b = "Bar"; }.c or "Xyzzzy"
```

```
{ a = "Foo"; b = "Bar"; }.c.d.e.f.g or "Xyzzzy"
```

will both evaluate to `"Xyzzzy"` because there is no `c` attribute in the set.

You can use arbitrary double-quoted strings as attribute names:

```
{ "$!@#?" = 123; }."$!@#?"
```

```
let bar = "bar"; in
{ "foo ${bar}" = 123; }."foo ${bar}"
```

Both will evaluate to `123`.

Attribute names support [string interpolation](#):

```
let bar = "foo"; in
{ foo = 123; }.${bar}
```

```
let bar = "foo"; in
{ ${bar} = 123; }.foo
```

Both will evaluate to `123`.

In the special case where an attribute name inside of a set declaration evaluates to `null` (which is normally an error, as `null` cannot be coerced to a string), that attribute is simply not added to the set:

```
{ ${if foo then "bar" else null} = true; }
```

This will evaluate to `{}` if `foo` evaluates to `false`.

A set that has a `__functor` attribute whose value is callable (i.e. is itself a function or a set with a `__functor` attribute whose value is callable) can be applied as if it were a function, with the set itself passed in first, e.g.,

```
let add = { __functor = self: x: x + self.x; };
  inc = add // { x = 1; };
in inc 1
```

evaluates to `2`. This can be used to attach metadata to a function without the caller needing to treat it specially, or to implement a form of object-oriented programming, for example.

Language Constructs

Recursive sets

Recursive sets are like normal [attribute sets](#), but the attributes can refer to each other.

```
rec-attrset = rec { [ name = expr ; ] ... }
```

Example:

```
rec {
  x = y;
  y = 123;
}.x
```

This evaluates to [123](#).

Note that without `rec` the binding `x = y;` would refer to the variable `y` in the surrounding scope, if one exists, and would be invalid if no such variable exists. That is, in a normal (non-recursive) set, attributes are not added to the lexical scope; in a recursive set, they are.

Recursive sets of course introduce the danger of infinite recursion. For example, the expression

```
rec {
  x = y;
  y = x;
}.x
```

will crash with an [infinite recursion encountered](#) error message.

Let-expressions

A let-expression allows you to define local variables for an expression.

```
let-in = let [ identifier = expr ]... in expr
```

Example:

```
let
  x = "foo";
  y = "bar";
in x + y
```

This evaluates to "foobar".

Inheriting attributes

When defining an [attribute set](#) or in a [let-expression](#) it is often convenient to copy variables from the surrounding lexical scope (e.g., when you want to propagate attributes). This can be shortened using the `inherit` keyword.

Example:

```
let x = 123; in
{
  inherit x;
  y = 456;
}
```

is equivalent to

```
let x = 123; in
{
  x = x;
  y = 456;
}
```

and both evaluate to `{ x = 123; y = 456; }`.

Note

This works because `x` is added to the lexical scope by the `let` construct.

It is also possible to inherit attributes from another attribute set.

Example:

In this fragment from `all-packages.nix`,

```
graphviz = (import ../../tools/graphics/graphviz) {
  inherit fetchurl stdenv libpng libjpeg expat x11 yacc;
  inherit (xorg) libXaw;
};

xorg = {
  libX11 = ...;
  libXaw = ...;
  ...
};

libpng = ...;
libjpg = ...;
...
```

the set used in the function call to the function defined in `../../tools/graphics/graphviz` inherits a number of variables from the surrounding scope (`fetchurl ... yacc`), but also inherits `libXaw` (the X Athena Widgets) from the `xorg` set.

Summarizing the fragment

```
...
inherit x y z;
inherit (src-set) a b c;
...
```

is equivalent to

```
...
x = x; y = y; z = z;
a = src-set.a; b = src-set.b; c = src-set.c;
...
```

when used while defining local variables in a let-expression or while defining a set.

Functions

Functions have the following form:

```
pattern: body
```

The pattern specifies what the argument of the function must look like, and binds variables in the body to (parts of) the argument. There are three kinds of patterns:

- If a pattern is a single identifier, then the function matches any argument. Example:

```
let negate = x: !x;
  concat = x: y: x + y;
in if negate true then concat "foo" "bar" else ""
```

Note that `concat` is a function that takes one argument and returns a function that takes another argument. This allows partial parameterisation (i.e., only filling some of the arguments of a function); e.g.,

```
map (concat "foo") [ "bar" "bla" "abc" ]
```

evaluates to `["foobar" "foobla" "fooabc"]`.

- A *set pattern* of the form `{ name1, name2, ..., nameN }` matches a set containing the listed attributes, and binds the values of those attributes to variables in the function body. For example, the function

```
{ x, y, z }: z + y + x
```

can only be called with a set containing exactly the attributes `x`, `y` and `z`. No other attributes are allowed. If you want to allow additional arguments, you can use an ellipsis (`...`):

```
{ x, y, z, ... }: z + y + x
```

This works on any set that contains at least the three named attributes.

It is possible to provide *default values* for attributes, in which case they are allowed to be missing. A default value is specified by writing `name ? e`, where `e` is an arbitrary expression. For example,

```
{ x, y ? "foo", z ? "bar" }: z + y + x
```

specifies a function that only requires an attribute named `x`, but optionally accepts `y` and `z`.

- An `@`-pattern provides a means of referring to the whole value being matched:

```
args@{ x, y, z, ... }: z + y + x + args.a
```

but can also be written as:

```
{ x, y, z, ... } @ args: z + y + x + args.a
```

Here `args` is bound to the argument *as passed*, which is further matched against the pattern `{ x, y, z, ... }`. The `@`-pattern makes mainly sense with an ellipsis(`...`) as you can access attribute names as `a`, using `args.a`, which was given as an additional attribute to the function.

Warning

`args@` binds the name `args` to the attribute set that is passed to the function. In particular, `args` does *not* include any default values specified with `?` in the function's set pattern.

For instance

```
let
  f = args@{ a ? 23, ... }: [ a args ];
in
  f {}
```

is equivalent to

```
let
  f = args @ { ... }: [ (args.a or 23) args ];
in
  f {}
```

and both expressions will evaluate to:

```
[ 23 {} ]
```

Note that functions do not have names. If you want to give them a name, you can bind them to an attribute, e.g.,

```
let concat = { x, y }: x + y;
in concat { x = "foo"; y = "bar"; }
```

Conditionals

Conditionals look like this:

```
if e1 then e2 else e3
```

where *e1* is an expression that should evaluate to a Boolean value (`true` or `false`).

Assertions

Assertions are generally used to check that certain requirements on or between features and dependencies hold. They look like this:

```
assert e1; e2
```

where *e1* is an expression that should evaluate to a Boolean value. If it evaluates to `true`, *e2* is returned; otherwise expression evaluation is aborted and a backtrace is printed.

Here is a Nix expression for the Subversion package that shows how assertions can be used::

```
{ localServer ? false
, httpServer ? false
, sslSupport ? false
, pythonBindings ? false
, javaSwigBindings ? false
, javahlBindings ? false
, stdenv, fetchurl
, openssl ? null, httpd ? null, db4 ? null, expat, swig ? null, j2sdk ? null
}:

assert localServer -> db4 != null; ①
assert httpServer -> httpd != null && httpd.expat == expat; ②
assert sslSupport -> openssl != null && (httpServer -> httpd.openssl ==
openssl); ③
assert pythonBindings -> swig != null && swig.pythonSupport;
assert javaSwigBindings -> swig != null && swig.javaSupport;
assert javahlBindings -> j2sdk != null;

stdenv.mkDerivation {
  name = "subversion-1.1.1";
  ...
  openssl = if sslSupport then openssl else null; ④
  ...
}
```

The points of interest are:

1. This assertion states that if Subversion is to have support for local repositories, then Berkeley DB is needed. So if the Subversion function is called with the `localServer` argument set to `true` but the `db4` argument set to `null`, then the evaluation fails.

Note that `->` is the [logical implication](#) Boolean operation.

2. This is a more subtle condition: if Subversion is built with Apache (`httpServer`) support, then the Expat library (an XML library) used by Subversion should be same as the one used by Apache. This is because in this configuration Subversion code ends up being linked with Apache code, and if the Expat libraries do not match, a build- or runtime link error or incompatibility might occur.
3. This assertion says that in order for Subversion to have SSL support (so that it can access `https` URLs), an OpenSSL library must be passed. Additionally, it says that *if* Apache support is enabled, then Apache's OpenSSL should match Subversion's. (Note that if Apache support is not enabled, we don't care about Apache's OpenSSL.)
4. The conditional here is not really related to assertions, but is worth pointing out: it ensures that if SSL support is disabled, then the Subversion derivation is not dependent on OpenSSL, even if a non-`null` value was passed. This prevents an unnecessary rebuild of Subversion if OpenSSL changes.

With-expressions

A *with-expression*,

```
with e1; e2
```

introduces the set *e1* into the lexical scope of the expression *e2*. For instance,

```
let as = { x = "foo"; y = "bar"; };
in with as; x + y
```

evaluates to `"foobar"` since the `with` adds the `x` and `y` attributes of `as` to the lexical scope in the expression `x + y`. The most common use of `with` is in conjunction with the `import` function. E.g.,

```
with (import ./definitions.nix); ...
```

makes all attributes defined in the file `definitions.nix` available as if they were defined locally in a `let`-expression.

The bindings introduced by `with` do not shadow bindings introduced by other means, e.g.

```
let a = 3; in with { a = 1; }; let a = 4; in with { a = 2; }; ...
```

establishes the same scope as

```
let a = 1; in let a = 2; in let a = 3; in let a = 4; in ...
```

Comments

Comments can be single-line, started with a `#` character, or inline/multi-line, enclosed within `/* ... */`.

String interpolation

String interpolation is a language feature where a [string](#), [path](#), or [attribute name](#) can contain expressions enclosed in `${ }` (dollar-sign with curly brackets).

Such a string is an *interpolated string*, and an expression inside is an *interpolated expression*.

Interpolated expressions must evaluate to one of the following:

- a [string](#)
- a [path](#)
- a [derivation](#)

Examples

String

Rather than writing

```
--with-freetype2-library=" + freetype + "/lib"
```

(where `freetype` is a [derivation](#)), you can instead write

```
--with-freetype2-library=${freetype}/lib"
```

The latter is automatically translated to the former.

A more complicated example (from the Nix expression for [Qt](#)):

```
configureFlags = "
-system-zlib -system-libpng -system-libjpeg
${if openglSupport then "-dlopen-opengl
-L${mesa}/lib -I${mesa}/include
-L${libXmu}/lib -I${libXmu}/include" else ""}
${if threadSupport then "-thread" else "-no-thread"}
";
```

Note that Nix expressions and strings can be arbitrarily nested; in this case the outer string contains various interpolated expressions that themselves contain strings (e.g., `"-thread"`), some of which in turn contain interpolated expressions (e.g., `${mesa}`).

Path

Rather than writing

```
./. + "/" + foo + "-" + bar + ".nix"
```

or

```
./. + "/${foo}-${bar}.nix"
```

you can instead write

```
./${foo}-${bar}.nix
```

Attribute name

Attribute names can be created dynamically with string interpolation:

```
let name = "foo"; in
{
  ${name} = "bar";
}

{ foo = "bar"; }
```

Operators

Name	Syntax	Associativity	Precedence
Attribute selection	<code>attrset . attrpath [or expr]</code>	none	1
Function application	<code>func expr</code>	left	2
Arithmetic negation	<code>- number</code>	none	3
Has attribute	<code>attrset ? attrpath</code>	none	4
List concatenation	<code>list ++ list</code>	right	5
Multiplication	<code>number ∗ number</code>	left	6
Division	<code>number / number</code>	left	6
Subtraction	<code>number - number</code>	left	7
Addition	<code>number + number</code>	left	7
String concatenation	<code>string + string</code>	left	7
Path concatenation	<code>path + path</code>	left	7
Path and string concatenation	<code>path + string</code>	left	7
String and path concatenation	<code>string + path</code>	left	7
Logical negation (NOT)	<code>! bool</code>	none	8
Update	<code>attrset // attrset</code>	right	9
Less than	<code>expr < expr</code>	none	10
Less than or equal to	<code>expr <= expr</code>	none	10
Greater than	<code>expr > expr</code>	none	10
Greater than or equal to	<code>expr >= expr</code>	none	10
Equality	<code>expr == expr</code>	none	11
Inequality	<code>expr != expr</code>	none	11
Logical conjunction (AND)	<code>bool && bool</code>	left	12
Logical disjunction (OR)	<code>bool bool</code>	left	13
Logical implication	<code>bool -> bool</code>	none	14

Attribute selection

attrset . *attrpath* [**or** *expr*]

Select the attribute denoted by attribute path *attrpath* from attribute set *attrset*. If the attribute doesn't exist, return the *expr* after **or** if provided, otherwise abort evaluation.

An attribute path is a dot-separated list of attribute names.

attrpath = *name* [. *name*]...

Has attribute

attrset ? *attrpath*

Test whether attribute set *attrset* contains the attribute denoted by *attrpath*. The result is a Boolean value.

Arithmetic

Numbers are type-compatible: Pure integer operations will always return integers, whereas any operation involving at least one floating point number return a floating point number.

See also [Comparison](#) and [Equality](#).

The **+** operator is overloaded to also work on strings and paths.

String concatenation

string **+** *string*

Concatenate two [strings](#) and merge their string contexts.

Path concatenation

path + *path*

Concatenate two [paths](#). The result is a path.

Path and string concatenation

path + *string*

Concatenate [path](#) with [string](#). The result is a path.

Note

The string must not have a string context that refers to a [store path](#).

String and path concatenation

string + *path*

Concatenate [string](#) with [path](#). The result is a string.

Important

The file or directory at *path* must exist and is copied to the [store](#). The path appears in the result as the corresponding [store path](#).

Update

attrset1 // *attrset2*

Update [attribute set](#) *attrset1* with names and values from *attrset2*.

The returned attribute set will have of all the attributes in *attrset1* and *attrset2*. If an attribute name is present in both, the attribute value from the latter is taken.

Comparison

Comparison is

- arithmetic for [numbers](#)
- lexicographic for [strings](#) and [paths](#)
- item-wise lexicographic for [lists](#): elements at the same index in both lists are compared according to their type and skipped if they are equal.

All comparison operators are implemented in terms of `<`, and the following equivalencies hold:

comparison	implementation
<code>a <= b</code>	<code>! (b < a)</code>
<code>a > b</code>	<code>b < a</code>
<code>a >= b</code>	<code>! (a < b)</code>

Equality

- [Attribute sets](#) and [lists](#) are compared recursively, and therefore are fully evaluated.
- Comparison of [functions](#) always returns `false`.
- Numbers are type-compatible, see [arithmetic](#) operators.
- Floating point numbers only differ up to a limited precision.

Logical implication

Equivalent to `! b1 || b2`.

Derivations

The most important built-in function is `derivation`, which is used to describe a single derivation (a build task). It takes as input a set, the attributes of which specify the inputs of the build.

- There must be an attribute named `system` whose value must be a string specifying a Nix system type, such as `"i686-linux"` or `"x86_64-darwin"`. (To figure out your system type, run `nix -vv --version`.) The build can only be performed on a machine and operating system matching the system type. (Nix can automatically [forward builds for other platforms](#) by forwarding them to other machines.)
- There must be an attribute named `name` whose value must be a string. This is used as a symbolic name for the package by `nix-env`, and it is appended to the output paths of the derivation.
- There must be an attribute named `builder` that identifies the program that is executed to perform the build. It can be either a derivation or a source (a local file reference, e.g., `./builder.sh`).
- Every attribute is passed as an environment variable to the builder. Attribute values are translated to environment variables as follows:
 - Strings and numbers are just passed verbatim.
 - A *path* (e.g., `../foo/sources.tar`) causes the referenced file to be copied to the store; its location in the store is put in the environment variable. The idea is that all sources should reside in the Nix store, since all inputs to a derivation should reside in the Nix store.
 - A *derivation* causes that derivation to be built prior to the present derivation; its default output path is put in the environment variable.
 - Lists of the previous types are also allowed. They are simply concatenated, separated by spaces.
 - `true` is passed as the string `1`, `false` and `null` are passed as an empty string.
- The optional attribute `args` specifies command-line arguments to be passed to the builder. It should be a list.
- The optional attribute `outputs` specifies a list of symbolic outputs of the derivation. By default, a derivation produces a single output path, denoted as `out`. However,

derivations can produce multiple output paths. This is useful because it allows outputs to be downloaded or garbage-collected separately. For instance, imagine a library package that provides a dynamic library, header files, and documentation. A program that links against the library doesn't need the header files and documentation at runtime, and it doesn't need the documentation at build time. Thus, the library package could specify:

```
outputs = [ "lib" "headers" "doc" ];
```

This will cause Nix to pass environment variables `lib`, `headers` and `doc` to the builder containing the intended store paths of each output. The builder would typically do something like

```
./configure \
--libdir=$lib/lib \
--includedir=$headers/include \
--docdir=$doc/share/doc
```

for an Autoconf-style package. You can refer to each output of a derivation by selecting it as an attribute, e.g.

```
buildInputs = [ pkg.lib pkg.headers ];
```

The first element of `outputs` determines the *default output*. Thus, you could also write

```
buildInputs = [ pkg pkg.headers ];
```

since `pkg` is equivalent to `pkg.lib`.

The function `mkDerivation` in the Nixpkgs standard environment is a wrapper around `derivation` that adds a default value for `system` and always uses Bash as the builder, to which the supplied builder is passed as a command-line argument. See the Nixpkgs manual for details.

The builder is executed as follows:

- A temporary directory is created under the directory specified by `TMPDIR` (default `/tmp`) where the build will take place. The current directory is changed to this directory.
- The environment is cleared and set to the derivation attributes, as specified above.

- In addition, the following variables are set:
 - `NIX_BUILD_TOP` contains the path of the temporary directory for this build.
 - Also, `TMPDIR`, `TEMPDIR`, `TMP`, `TEMP` are set to point to the temporary directory. This is to prevent the builder from accidentally writing temporary files anywhere else. Doing so might cause interference by other processes.
 - `PATH` is set to `/path-not-set` to prevent shells from initialising it to their built-in default value.
 - `HOME` is set to `/homeless-shelter` to prevent programs from using `/etc/passwd` or the like to find the user's home directory, which could cause impurity. Usually, when `HOME` is set, it is used as the location of the home directory, even if it points to a non-existent path.
 - `NIX_STORE` is set to the path of the top-level Nix store directory (typically, `/nix/store`).
 - For each output declared in `outputs`, the corresponding environment variable is set to point to the intended path in the Nix store for that output. Each output path is a concatenation of the cryptographic hash of all build inputs, the `name` attribute and the output name. (The output name is omitted if it's `out`.)
- If an output path already exists, it is removed. Also, locks are acquired to prevent multiple Nix instances from performing the same build at the same time.
- A log of the combined standard output and error is written to `/nix/var/log/nix`.
- The builder is executed with the arguments specified by the attribute `args`. If it exits with exit code 0, it is considered to have succeeded.
- The temporary directory is removed (unless the `-K` option was specified).
- If the build was successful, Nix scans each output path for references to input paths by looking for the hash parts of the input paths. Since these are potential runtime dependencies, Nix registers them as dependencies of the output paths.
- After the build, Nix sets the last-modified timestamp on all files in the build result to 1 (00:00:01 1/1/1970 UTC), sets the group to the default group, and sets the mode of the file to 0444 or 0555 (i.e., read-only, with execute permission enabled if the file was originally executable). Note that possible `setuid` and `setgid` bits are cleared. Setuid and setgid programs are not currently supported by Nix. This is because the Nix archives used in deployment have no concept of ownership information, and because

it makes the build result dependent on the user performing the build.

Advanced Attributes

Derivations can declare some infrequently used optional attributes.

- `allowedReferences`

The optional attribute `allowedReferences` specifies a list of legal references (dependencies) of the output of the builder. For example,

```
allowedReferences = [];
```

enforces that the output of a derivation cannot have any runtime dependencies on its inputs. To allow an output to have a runtime dependency on itself, use `"out"` as a list item. This is used in NixOS to check that generated files such as initial ramdisks for booting Linux don't have accidental dependencies on other paths in the Nix store.

- `allowedRequisites`

This attribute is similar to `allowedReferences`, but it specifies the legal requisites of the whole closure, so all the dependencies recursively. For example,

```
allowedRequisites = [ foobar ];
```

enforces that the output of a derivation cannot have any other runtime dependency than `foobar`, and in addition it enforces that `foobar` itself doesn't introduce any other dependency itself.

- `disallowedReferences`

The optional attribute `disallowedReferences` specifies a list of illegal references (dependencies) of the output of the builder. For example,

```
disallowedReferences = [ foo ];
```

enforces that the output of a derivation cannot have a direct runtime dependencies on the derivation `foo`.

- `disallowedRequisites`

This attribute is similar to `disallowedReferences`, but it specifies illegal requisites for the whole closure, so all the dependencies recursively. For example,

```
disallowedRequisites = [ foobar ];
```

enforces that the output of a derivation cannot have any runtime dependency on `foobar` or any other derivation depending recursively on `foobar`.

- **`exportReferencesGraph`**

This attribute allows builders access to the references graph of their inputs. The attribute is a list of inputs in the Nix store whose references graph the builder needs to know. The value of this attribute should be a list of pairs `[name1 path1 name2 path2 ...]`. The references graph of each `pathN` will be stored in a text file `nameN` in the temporary build directory. The text files have the format used by `nix-store --register-validity` (with the `deriver` fields left empty). For example, when the following derivation is built:

```
derivation {  
  ...  
  exportReferencesGraph = [ "libfoo-graph" libfoo ];  
};
```

the references graph of `libfoo` is placed in the file `libfoo-graph` in the temporary build directory.

`exportReferencesGraph` is useful for builders that want to do something with the closure of a store path. Examples include the builders in NixOS that generate the initial ramdisk for booting Linux (a `cpio` archive containing the closure of the boot script) and the ISO-9660 image for the installation CD (which is populated with a Nix store containing the closure of a bootable NixOS configuration).

- **`impureEnvVars`**

This attribute allows you to specify a list of environment variables that should be passed from the environment of the calling user to the builder. Usually, the environment is cleared completely when the builder is executed, but with this attribute you can allow specific environment variables to be passed unmodified. For example, `fetchurl` in Nixpkgs has the line

```
impureEnvVars = [ "http_proxy" "https_proxy" ... ];
```

to make it use the proxy server configuration specified by the user in the environment variables `http_proxy` and friends.

This attribute is only allowed in *fixed-output derivations* (see below), where impurities such as these are okay since (the hash of) the output is known in advance. It is ignored for all other derivations.

Warning

`impureEnvVars` implementation takes environment variables from the current builder process. When a daemon is building its environmental variables are used. Without the daemon, the environmental variables come from the environment of the `nix-build`.

- `outputHash`; `outputHashAlgo`; `outputHashMode`

These attributes declare that the derivation is a so-called *fixed-output derivation*, which means that a cryptographic hash of the output is already known in advance. When the build of a fixed-output derivation finishes, Nix computes the cryptographic hash of the output and compares it to the hash declared with these attributes. If there is a mismatch, the build fails.

The rationale for fixed-output derivations is derivations such as those produced by the `fetchurl` function. This function downloads a file from a given URL. To ensure that the downloaded file has not been modified, the caller must also specify a cryptographic hash of the file. For example,

```
fetchurl {  
    url = "http://ftp.gnu.org/pub/gnu/hello/hello-2.1.1.tar.gz";  
    sha256 = "1md7jsfd8pa45z73bz1kszpp01yw6x5ljkjk2hx7wl800any6465";  
}
```

It sometimes happens that the URL of the file changes, e.g., because servers are reorganised or no longer available. We then must update the call to `fetchurl`, e.g.,

```
fetchurl {  
    url = "ftp://ftp.nluug.nl/pub-gnu/hello/hello-2.1.1.tar.gz";  
    sha256 = "1md7jsfd8pa45z73bz1kszpp01yw6x5ljkjk2hx7wl800any6465";  
}
```

If a `fetchurl` derivation was treated like a normal derivation, the output paths of the derivation and *all derivations depending on it* would change. For instance, if we were to change the URL of the Glibc source distribution in Nixpkgs (a package on which almost all other packages depend) massive rebuilds would be needed. This is unfortunate for a change which we know cannot have a real effect as it propagates upwards through the dependency graph.

For fixed-output derivations, on the other hand, the name of the output path only

depends on the `outputHash*` and `name` attributes, while all other attributes are ignored for the purpose of computing the output path. (The `name` attribute is included because it is part of the path.)

As an example, here is the (simplified) Nix expression for `fetchurl`:

```
{ stdenv, curl }: # The curl program is used for downloading.

{ url, sha256 }:

stdenv.mkDerivation {
  name = baseNameOf (toString url);
  builder = ./builder.sh;
  buildInputs = [ curl ];

  # This is a fixed-output derivation; the output must be a regular
  # file with SHA256 hash sha256.
  outputHashMode = "flat";
  outputHashAlgo = "sha256";
  outputHash = sha256;

  inherit url;
}
```

The `outputHashAlgo` attribute specifies the hash algorithm used to compute the hash. It can currently be `"sha1"`, `"sha256"` or `"sha512"`.

The `outputHashMode` attribute determines how the hash is computed. It must be one of the following two values:

- `"flat"`

The output must be a non-executable regular file. If it isn't, the build fails. The hash is simply computed over the contents of that file (so it's equal to what Unix commands like `sha256sum` or `sha1sum` produce).

This is the default.

- `"recursive"`

The hash is computed over the NAR archive dump of the output (i.e., the result of `nix-store --dump`). In this case, the output can be anything, including a directory tree.

The `outputHash` attribute, finally, must be a string containing the hash in either hexadecimal or base-32 notation. (See the [nix-hash command](#) for information about converting to and from base-32 notation.)

- [__contentAddressed](#)
-

Warning This attribute is part of an [experimental feature](#).

To use this attribute, you must enable the `ca-derivations` experimental feature. For example, in `nix.conf` you could add:

```
extra-experimental-features = ca-derivations
```

If this attribute is set to `true`, then the derivation outputs will be stored in a content-addressed location rather than the traditional input-addressed one.

Setting this attribute also requires setting `outputHashMode` and `outputHashAlgo` like for *fixed-output derivations* (see above).

- [passAsFile](#)

A list of names of attributes that should be passed via files rather than environment variables. For example, if you have

```
passAsFile = ["big"];
big = "a very long string";
```

then when the builder runs, the environment variable `bigPath` will contain the absolute path to a temporary file containing `a very long string`. That is, for any attribute `x` listed in `passAsFile`, Nix will pass an environment variable `xPath` holding the path of the file containing the value of attribute `x`. This is useful when you need to pass large strings to a builder, since most operating systems impose a limit on the size of the environment (typically, a few hundred kilobyte).

- [preferLocalBuild](#)

If this attribute is set to `true` and [distributed building is enabled](#), then, if possible, the derivation will be built locally instead of forwarded to a remote machine. This is appropriate for trivial builders where the cost of doing a download or remote build would exceed the cost of building locally.

- [allowSubstitutes](#)

If this attribute is set to `false`, then Nix will always build this derivation; it will not try to substitute its outputs. This is useful for very trivial derivations (such as `writeText` in Nixpkgs) that are cheaper to build than to substitute from a binary cache.

Note

You need to have a builder configured which satisfies the derivation's `system` attribute, since the derivation cannot be substituted. Thus it is usually a good idea to align `system` with `builtins.currentSystem` when setting `allowSubstitutes` to `false`. For most trivial derivations this should be the case.

- [`_structuredAttrs`](#)

If the special attribute `_structuredAttrs` is set to `true`, the other derivation attributes are serialised in JSON format and made available to the builder via the file `.attrs.json` in the builder's temporary directory. This obviates the need for `passAsFile` since JSON files have no size restrictions, unlike process environments.

It also makes it possible to tweak derivation settings in a structured way; see `outputChecks` for example.

As a convenience to Bash builders, Nix writes a script named `.attrs.sh` to the builder's directory that initialises shell variables corresponding to all attributes that are representable in Bash. This includes non-nested (associative) arrays. For example, the attribute `hardening.format = true` ends up as the Bash associative array element `${hardening[format]}` .

- [`outputChecks`](#)

When using [structured attributes](#), the `outputChecks` attribute allows defining checks per-output.

In addition to `allowedReferences`, `allowedRequisites`, `disallowedReferences` and `disallowedRequisites`, the following attributes are available:

- `maxSize` defines the maximum size of the resulting [store object](#).
- `maxClosureSize` defines the maximum size of the output's closure.
- `ignoreSelfRefs` controls whether self-references should be considered when checking for allowed references/requisites.

Example:

```
__structuredAttrs = true;

outputChecks.out = {
  # The closure of 'out' must not be larger than 256 MiB.
  maxClosureSize = 256 * 1024 * 1024;

  # It must not refer to the C compiler or to the 'dev' output.
  disallowedRequisites = [ stdenv.cc "dev" ];
};

outputChecks.dev = {
  # The 'dev' output must not be larger than 128 KiB.
  maxSize = 128 * 1024;
};
```

- [unsafeDiscardReferences](#)

When using [structured attributes](#), the attribute `unsafeDiscardReferences` is an attribute set with a boolean value for each output name. If set to `true`, it disables scanning the output for runtime dependencies.

Example:

```
__structuredAttrs = true;
unsafeDiscardReferences.out = true;
```

This is useful, for example, when generating self-contained filesystem images with their own embedded Nix store: hashes found inside such an image refer to the embedded store and not to the host's Nix store.

Built-in Constants

These constants are built into the Nix language evaluator:

`builtins` (set)

Contains all the [built-in functions](#) and values.

Since built-in functions were added over time, [testing for attributes](#) in `builtins` can be used for graceful fallback on older Nix installations:

```
# if hasContext is not available, we assume `s` has a context
if builtins ? hasContext then builtins.hasContext s else true
```

`currentSystem` (string)

The value of the [system configuration option](#).

It can be used to set the `system` attribute for `builtins.derivation` such that the resulting derivation can be built on the same system that evaluates the Nix expression:

```
builtins.derivation {
  # ...
  system = builtins.currentSystem;
}
```

It can be overridden in order to create derivations for different system than the current one:

```
$ nix-instantiate --system "mips64-linux" --eval --expr
'builtins.currentSystem'
"mips64-linux"
```

Note

Not available in [pure evaluation mode](#).

`currentTime` (integer)

Return the [Unix time](#) at first evaluation. Repeated references to that name will re-use the initially obtained value.

Example:

```
$ nix repl  
Welcome to Nix 2.15.1 Type :? for help.  
  
nix-repl> builtins.currentTime  
1683705525  
  
nix-repl> builtins.currentTime  
1683705525
```

The [store path](#) of a derivation depending on `currentTime` will differ for each evaluation, unless both evaluate `builtins.currentTime` in the same second.

Note

Not available in [pure evaluation mode](#).

`false` (Boolean)

Primitive value.

It can be returned by [comparison operators](#) and used in [conditional expressions](#).

The name `false` is not special, and can be shadowed:

```
nix-repl> let false = 1; in false  
1
```

`langVersion` (integer)

The current version of the Nix language.

`nixPath` (list)

The search path used to resolve angle bracket path lookups.

Angle bracket expressions can be [desugared](#) using this and `builtins.findFile`:

```
<nixpkgs>
```

is equivalent to:

```
builtins.findFile builtins.nixPath "nixpkgs"
```

nixVersion (string)

The version of Nix.

For example, where the command line returns the current Nix version,

```
$ nix --version
nix (Nix) 2.16.0
```

the Nix language evaluator returns the same value:

```
nix-repl> builtins.nixVersion
"2.16.0"
```

null (null)

Primitive value.

The name `null` is not special, and can be shadowed:

```
nix-repl> let null = 1; in null
1
```

storeDir (string)

Logical file system location of the [Nix store](#) currently in use.

This value is determined by the `store` parameter in [Store URLs](#):

```
$ nix-instantiate --store 'dummy://?store=/blah' --eval --expr
builtins.storeDir
"/blah"
```

true (Boolean)

Primitive value.

It can be returned by [comparison operators](#) and used in [conditional expressions](#).

The name `true` is not special, and can be shadowed:

```
nix-repl> let true = 1; in true  
1
```

Built-in Functions

This section lists the functions built into the Nix language evaluator. All built-in functions are available through the global `builtins` constant.

For convenience, some built-ins can be accessed directly:

- `derivation`
- `import`
- `abort`
- `throw`

`derivation attrs`

`derivation` is described in [its own section](#).

`abort s`

Abort Nix expression evaluation and print the error message `s`.

`add e1 e2`

Return the sum of the numbers `e1` and `e2`.

`all pred list`

Return `true` if the function `pred` returns `true` for all elements of `list`, and `false` otherwise.

`any pred list`

Return `true` if the function `pred` returns `true` for at least one element of `list`, and `false` otherwise.

`attrNames set`

Return the names of the attributes in the set `set` in an alphabetically sorted list. For instance, `builtins.attrNames { y = 1; x = "foo"; }` evaluates to `["x" "y"]`.

`attrValues set`

Return the values of the attributes in the set `set` in the order corresponding to the sorted attribute names.

`baseNameOf s`

Return the *base name* of the string *s*, that is, everything following the final slash in the string. This is similar to the GNU `basename` command.

`bitAnd e1 e2`

Return the bitwise AND of the integers *e1* and *e2*.

`bitOr e1 e2`

Return the bitwise OR of the integers *e1* and *e2*.

`bitXor e1 e2`

Return the bitwise XOR of the integers *e1* and *e2*.

`break v`

In debug mode (enabled using `--debugger`), pause Nix expression evaluation and enter the REPL. Otherwise, return the argument *v*.

`catAttrs attr list`

Collect each attribute named *attr* from a list of attribute sets. Attrsets that don't contain the named attribute are ignored. For example,

```
builtins.catAttrs "a" [{a = 1;} {b = 0;} {a = 2;}]
```

evaluates to `[1 2]`.

`ceil double`

Converts an IEEE-754 double-precision floating-point number (*double*) to the next higher integer.

If the datatype is neither an integer nor a "float", an evaluation error will be thrown.

`compareVersions s1 s2`

Compare two strings representing versions and return `-1` if version *s1* is older than version *s2*, `0` if they are the same, and `1` if *s1* is newer than *s2*. The version comparison algorithm is the same as the one used by `nix-env -u`.

`concatLists lists`

Concatenate a list of lists into a single list.

`concatMap f list`

This function is equivalent to `builtins.concatLists (map f list)` but is more efficient.

`concatStringsSep separator list`

Concatenate a list of strings with a separator between each element, e.g.

```
concatStringsSep "/" ["usr" "local" "bin"] == "usr/local/bin".
```

`deepSeq e1 e2`

This is like `seq e1 e2`, except that `e1` is evaluated *deeply*: if it's a list or set, its elements or attributes are also evaluated recursively.

`dirOf s`

Return the directory part of the string `s`, that is, everything before the final slash in the string. This is similar to the GNU `dirname` command.

`div e1 e2`

Return the quotient of the numbers `e1` and `e2`.

`elem x xs`

Return `true` if a value equal to `x` occurs in the list `xs`, and `false` otherwise.

`elemAt xs n`

Return element `n` from the list `xs`. Elements are counted starting from 0. A fatal error occurs if the index is out of bounds.

`fetchClosure args`

Fetch a store path `closure` from a binary cache, and return the store path as a string with context.

This function can be invoked in three ways, that we will discuss in order of preference.

Fetch a content-addressed store path

Example:

```
builtins.fetchClosure {  
  fromStore = "https://cache.nixos.org";  
  fromPath = /nix/store/ldbh1whh39wha58rm61bkiiwm6j7211j-git-2.33.1;  
}
```

This is the simplest invocation, and it does not require the user of the expression to configure [trusted-public-keys](#) to ensure their authenticity.

If your store path is [input addressed](#) instead of content addressed, consider the other two invocations.

Fetch any store path and rewrite it to a fully content-addressed store path

Example:

```
builtins.fetchClosure {  
  fromStore = "https://cache.nixos.org";  
  fromPath = /nix/store/r2jd6ygnmirm2g803mksqqjm4y39yi6i-git-2.33.1;  
  toPath = /nix/store/ldbh1whh39wha58rm61bkiiwm6j7211j-git-2.33.1;  
}
```

This example fetches `/nix/store/r2jd...` from the specified binary cache, and rewrites it into the content-addressed store path `/nix/store/ldbh....`.

Like the previous example, no extra configuration or privileges are required.

To find out the correct value for `toPath` given a `fromPath`, use [nix store make-content-addressed](#):

```
# nix store make-content-addressed --from https://cache.nixos.org  
/nix/store/r2jd6ygnmirm2g803mksqqjm4y39yi6i-git-2.33.1  
rewrote '/nix/store/r2jd6ygnmirm2g803mksqqjm4y39yi6i-git-2.33.1' to  
'/nix/store/ldbh1whh39wha58rm61bkiiwm6j7211j-git-2.33.1'
```

Alternatively, set `toPath = ""` and find the correct `toPath` in the error message.

Fetch an input-addressed store path as is

Example:

```
builtins.fetchClosure {  
  fromStore = "https://cache.nixos.org";  
  fromPath = /nix/store/r2jd6ygnmirm2g803mksqqjm4y39yi6i-git-2.33.1;  
  inputAddressed = true;  
}
```

It is possible to fetch an [input-addressed store path](#) and return it as is. However, this is the least preferred way of invoking `fetchClosure`, because it requires that the input-addressed paths are trusted by the Nix configuration.

`builtins.storePath`

`fetchClosure` is similar to `builtins.storePath` in that it allows you to use a previously built store path in a Nix expression. However, `fetchClosure` is more reproducible because it specifies a binary cache from which the path can be fetched. Also, using content-addressed store paths does not require users to configure [trusted-public-keys](#) to ensure their authenticity.

This function is only available if the [fetch-closure](#) experimental feature is enabled.

`fetchGit args`

Fetch a path from git. `args` can be a URL, in which case the HEAD of the repo at that URL is fetched. Otherwise, it can be an attribute with the following attributes (all except `url` optional):

- `url`

The URL of the repo.

- `name` (default: *basename of the URL*)

The name of the directory the repo should be exported to in the store.

- `rev` (default: *the tip of ref*)

The [Git revision](#) to fetch. This is typically a commit hash.

- `ref` (default: `HEAD`)

The [Git reference](#) under which to look for the requested revision. This is often a branch or tag name.

By default, the `ref` value is prefixed with `refs/heads/`. As of 2.3.0, Nix will not prefix `refs/heads/` if `ref` starts with `refs/`.

- `submodules` (default: `false`)

A Boolean parameter that specifies whether submodules should be checked out.

- `shallow` (default: `false`)

A Boolean parameter that specifies whether fetching from a shallow remote repository is allowed. This still performs a full clone of what is available on the remote.

- `allRefs`

Whether to fetch all references of the repository. With this argument being true, it's possible to load a `rev` from *any* `ref` (by default only `rev`s from the specified `ref` are supported).

Here are some examples of how to use `fetchGit`.

- To fetch a private repository over SSH:

```
builtins.fetchGit {  
    url = "git@github.com:my-secret/repository.git";  
    ref = "master";  
    rev = "adab8b916a45068c044658c4158d81878f9ed1c3";  
}
```

- To fetch an arbitrary reference:

```
builtins.fetchGit {  
    url = "https://github.com/NixOS/nix.git";  
    ref = "refs/heads/0.5-release";  
}
```

- If the revision you're looking for is in the default branch of the git repository you don't strictly need to specify the branch name in the `ref` attribute.

However, if the revision you're looking for is in a future branch for the non-default branch you will need to specify the the `ref` attribute as well.

```
builtins.fetchGit {  
    url = "https://github.com/nixos/nix.git";  
    rev = "841fcbd04755c7a2865c51c1e2d3b045976b7452";  
    ref = "1.11-maintenance";  
}
```

Note

It is nice to always specify the branch which a revision belongs to. Without the branch being specified, the fetcher might fail if the default branch changes. Additionally, it can be confusing to try a commit from a non-default branch and see the fetch fail. If the branch is specified the fault is much more obvious.

- If the revision you're looking for is in the default branch of the git repository you may omit the `ref` attribute.

```
builtins.fetchGit {  
    url = "https://github.com/nixos/nix.git";  
    rev = "841fcbd04755c7a2865c51c1e2d3b045976b7452";  
}
```

- To fetch a specific tag:

```
builtins.fetchGit {  
    url = "https://github.com/nixos/nix.git";  
    ref = "refs/tags/1.9";  
}
```

- To fetch the latest version of a remote branch:

```
builtins.fetchGit {  
    url = "ssh://git@github.com/nixos/nix.git";  
    ref = "master";  
}
```

Nix will refetch the branch according to the `tarball-ttl` setting.

This behavior is disabled in [pure evaluation mode](#).

- To fetch the content of a checked-out work directory:

```
builtins.fetchGit ./work-dir
```

If the URL points to a local directory, and no `ref` or `rev` is given, `fetchGit` will use the current content of the checked-out files, even if they are not committed or added to Git's index. It will only consider files added to the Git repository, as listed by `git ls-files`.

fetchTarball args

Download the specified URL, unpack it and return the path of the unpacked tree. The file must be a tape archive (`.tar`) compressed with `gzip`, `bzip2` or `xz`. The top-level path component of the files in the tarball is removed, so it is best if the tarball contains a single directory at top level. The typical use of the function is to obtain external Nix expression dependencies, such as a particular version of Nixpkgs, e.g.

```
with import (fetchTarball https://github.com/NixOS/nixpkgs/archive/nixos-14.12.tar.gz) {};
stdenv.mkDerivation { ... }
```

The fetched tarball is cached for a certain amount of time (1 hour by default) in `~/.cache/nix/tarballs/`. You can change the cache timeout either on the command line with `--tarball-ttl number-of-seconds` or in the Nix configuration file by adding the line `tarball-ttl = number-of-seconds`.

Note that when obtaining the hash with `nix-prefetch-url` the option `--unpack` is required.

This function can also verify the contents against a hash. In that case, the function takes a set instead of a URL. The set requires the attribute `url` and the attribute `sha256`, e.g.

```
with import (fetchTarball {
  url = "https://github.com/NixOS/nixpkgs/archive/nixos-14.12.tar.gz";
  sha256 = "1jppksrfvbk5ypiqdz4cddxdl8z6zyzdb2srq8fcffr327ld5jj2";
}) {};
stdenv.mkDerivation { ... }
```

Not available in [restricted evaluation mode](#).

fetchurl url

Download the specified URL and return the path of the downloaded file.

Not available in [restricted evaluation mode](#).

filter f list

Return a list consisting of the elements of `list` for which the function `f` returns `true`.

filterSource e1 e2

Warning

`filterSource` should not be used to filter store paths. Since `filterSource` uses the name of the input directory while naming the output directory, doing so will produce a directory name in the form of `<hash2>-<hash>-<name>`, where `<hash>-<name>` is the name of the input directory. Since `<hash>` depends on the unfiltered directory, the name of the output directory will indirectly depend on files that are filtered out by the function. This will trigger a rebuild even when a filtered out file is changed. Use `builtins.path` instead, which allows specifying the name of the output directory.

This function allows you to copy sources into the Nix store while filtering certain files. For instance, suppose that you want to use the directory `source-dir` as an input to a Nix expression, e.g.

```
stdenv.mkDerivation {  
  ...  
  src = ./source-dir;  
}
```

However, if `source-dir` is a Subversion working copy, then all those annoying `.svn` subdirectories will also be copied to the store. Worse, the contents of those directories may change a lot, causing lots of spurious rebuilds. With `filterSource` you can filter out the `.svn` directories:

```
src = builtins.filterSource  
(path: type: type != "directory" || basePath path != ".svn")  
./source-dir;
```

Thus, the first argument `e1` must be a predicate function that is called for each regular file, directory or symlink in the source tree `e2`. If the function returns `true`, the file is copied to the Nix store, otherwise it is omitted. The function is called with two arguments. The first is the full path of the file. The second is a string that identifies the type of the file, which is either `"regular"`, `"directory"`, `"symlink"` or `"unknown"` (for other kinds of files such as device nodes or fifos — but note that those cannot be copied to the Nix store, so if the predicate returns `true` for them, the copy will fail). If you exclude a directory, the entire corresponding subtree of `e2` will be excluded.

`findFile search path lookup path`

Look up the given path with the given search path.

A search path is represented list of [attribute sets](#) with two attributes, `prefix`, and `path`. `prefix` is a relative path. `path` denotes a file system location; the exact syntax depends on the command line interface.

Examples of search path attribute sets:

- {
 `prefix = "nixos-config";`
 `path = "/etc/nixos/configuration.nix";`
}

- {
 `prefix = "";`
 `path = "/nix/var/nix/profiles/per-user/root/channels";`
}

The lookup algorithm checks each entry until a match is found, returning a [path value](#) of the match.

This is the process for each entry: If the lookup path matches `prefix`, then the remainder of the lookup path (the "suffix") is searched for within the directory denoted by `path`. Note that the `path` may need to be downloaded at this point to look inside. If the suffix is found inside that directory, then the entry is a match; the combined absolute path of the directory (now downloaded if need be) and the suffix is returned.

The syntax

```
<nixpkgs>
```

is equivalent to:

```
builtins.findFile builtins.nixPath "nixpkgs"
```

```
flakeRefToString attrs
```

Convert a flake reference from attribute set format to URL format.

For example:

```
builtins.flakeRefToString {
  dir = "lib"; owner = "NixOS"; ref = "23.05"; repo = "nixpkgs"; type =
"github";
}
```

evaluates to

```
"github:NixOS/nixpkgs/23.05?dir=lib"
```

This function is only available if the [flakes](#) experimental feature is enabled.

`floor double`

Converts an IEEE-754 double-precision floating-point number (*double*) to the next lower integer.

If the datatype is neither an integer nor a "float", an evaluation error will be thrown.

`foldl' op nul list`

Reduce a list by applying a binary operator, from left to right, e.g. `foldl' op nul [x0 x1 x2 ...] = op (op (op nul x0) x1) x2) ...`. For example, `foldl' (x: y: x + y) 0 [1 2 3]` evaluates to 6. The return value of each application of `op` is evaluated immediately, even for intermediate values.

`fromJSON e`

Convert a JSON string to a Nix value. For example,

```
builtins.fromJSON '{"x": [1, 2, 3], "y": null}'
```

returns the value `{ x = [1 2 3]; y = null; }`.

`fromTOML e`

Convert a TOML string to a Nix value. For example,

```
builtins.fromTOML """
x=1
s="a"
[table]
y=2
""
```

returns the value `{ s = "a"; table = { y = 2; }; x = 1; }`.

functionArgs *f*

Return a set containing the names of the formal arguments expected by the function *f*. The value of each attribute is a Boolean denoting whether the corresponding argument has a default value. For instance, `functionArgs ({ x, y ? 123}: ...) = { x = false; y = true; }`.

"Formal argument" here refers to the attributes pattern-matched by the function. Plain lambdas are not included, e.g. `functionArgs (x: ...) = { }`.

genList *generator length*

Generate list of size *length*, with each element *i* equal to the value returned by *generator i*. For example,

```
builtins.genList (x: x * x) 5
```

returns the list `[0 1 4 9 16]`.

genericClosure *attrset*

Take an *attrset* with values named `startSet` and `operator` in order to return a *list of attrsets* by starting with the `startSet` and recursively applying the `operator` function to each `item`. The *attrsets* in the `startSet` and the *attrsets* produced by `operator` must contain a value named `key` which is comparable. The result is produced by calling `operator` for each `item` with a value for `key` that has not been called yet including newly produced `item`s. The function terminates when no new `item`s are produced. The resulting *list of attrsets* contains only *attrsets* with a unique key. For example,

```
builtins.genericClosure {
  startSet = [ {key = 5;} ];
  operator = item: [{
    key = if (item.key / 2) * 2 == item.key
      then item.key / 2
      else 3 * item.key + 1;
  }];
}
```

evaluates to

```
[ { key = 5; } { key = 16; } { key = 8; } { key = 4; } { key = 2; } { key = 1; } ]
```

getAttr s set

`getAttr` returns the attribute named `s` from `set`. Evaluation aborts if the attribute doesn't exist. This is a dynamic version of the `.` operator, since `s` is an expression rather than an identifier.

getContext s

Return the string context of `s`.

The string context tracks references to derivations within a string. It is represented as an attribute set of [store derivation](#) paths mapping to output names.

Using [string interpolation](#) on a derivation will add that derivation to the string context. For example,

```
builtins.getContext "${derivation { name = "a"; builder = "b"; system = "c"; }}"
```

evaluates to

```
{ "/nix/store/arhvjaf6zmlyn8vh8fgn55rpwnxq0n7l-a.drv" = { outputs = [ "out" ]; }; }
```

getEnv s

`getEnv` returns the value of the environment variable `s`, or an empty string if the variable doesn't exist. This function should be used with care, as it can introduce all sorts of nasty environment dependencies in your Nix expression.

`getEnv` is used in Nix Packages to locate the file `~/.nixpkgs/config.nix`, which contains user-local settings for Nix Packages. (That is, it does a `getEnv "HOME"` to locate the user's home directory.)

getFlake args

Fetch a flake from a flake reference, and return its output attributes and some metadata. For example:

```
(builtins.getFlake  
"nix/55bc52401966fbffa525c574c14f67b00bc4fb3a").packages.x86_64-linux.nix
```

Unless impure evaluation is allowed (`--impure`), the flake reference must be "locked", e.g. contain a Git revision or content hash. An example of an unlocked usage is:

```
(builtins.getFlake "github:edolstra/dwarfss").rev
```

This function is only available if the [flakes](#) experimental feature is enabled.

groupBy *f* *list*

Groups elements of *list* together by the string returned from the function *f* called on each element. It returns an attribute set where each attribute value contains the elements of *list* that are mapped to the same corresponding attribute name returned by *f*.

For example,

```
builtins.groupBy (builtins.substring 0 1) ["foo" "bar" "baz"]
```

evaluates to

```
{ b = [ "bar" "baz" ]; f = [ "foo" ]; }
```

hasAttr *s* *set*

`hasAttr` returns `true` if *set* has an attribute named *s*, and `false` otherwise. This is a dynamic version of the `?` operator, since *s* is an expression rather than an identifier.

hasContext *s*

Return `true` if string *s* has a non-empty context. The context can be obtained with `getContext`.

hashFile *type* *p*

Return a base-16 representation of the cryptographic hash of the file at path *p*. The hash algorithm specified by *type* must be one of `"md5"`, `"sha1"`, `"sha256"` or `"sha512"`.

hashString *type* *s*

Return a base-16 representation of the cryptographic hash of string *s*. The hash algorithm specified by *type* must be one of `"md5"`, `"sha1"`, `"sha256"` or `"sha512"`.

head *list*

Return the first element of a list; abort evaluation if the argument isn't a list or is an empty list. You can test whether a list is empty by comparing it with `[]`.

import path

Load, parse and return the Nix expression in the file *path*.

The value *path* can be a path, a string, or an attribute set with an `_toString` attribute or a `outPath` attribute (as derivations or flake inputs typically have).

If *path* is a directory, the file `default.nix` in that directory is loaded.

Evaluation aborts if the file doesn't exist or contains an incorrect Nix expression.

`import` implements Nix's module system: you can put any Nix expression (such as a set or a function) in a separate file, and use it from Nix expressions in other files.

Note

Unlike some languages, `import` is a regular function in Nix. Paths using the angle bracket syntax (e.g., `import <foo>`) are normal `path values`.

A Nix expression loaded by `import` must not contain any *free variables* (identifiers that are not defined in the Nix expression itself and are not built-in). Therefore, it cannot refer to variables that are in scope at the call site. For instance, if you have a calling expression

```
rec {  
  x = 123;  
  y = import ./foo.nix;  
}
```

then the following `foo.nix` will give an error:

```
x + 456
```

since `x` is not in scope in `foo.nix`. If you want `x` to be available in `foo.nix`, you should pass it as a function argument:

```
rec {  
  x = 123;  
  y = import ./foo.nix x;  
}
```

and

```
x: x + 456
```

(The function argument doesn't have to be called `x` in `foo.nix`; any name would work.)

`intersectAttrs e1 e2`

Return a set consisting of the attributes in the set `e2` which have the same name as some attribute in `e1`.

Performs in $O(n \log m)$ where n is the size of the smaller set and m the larger set's size.

`isAttrs e`

Return `true` if `e` evaluates to a set, and `false` otherwise.

`isBool e`

Return `true` if `e` evaluates to a bool, and `false` otherwise.

`isFloat e`

Return `true` if `e` evaluates to a float, and `false` otherwise.

`isFunction e`

Return `true` if `e` evaluates to a function, and `false` otherwise.

`isInt e`

Return `true` if `e` evaluates to an integer, and `false` otherwise.

`isList e`

Return `true` if `e` evaluates to a list, and `false` otherwise.

`isNull e`

Return `true` if `e` evaluates to `null`, and `false` otherwise.

Warning

This function is *deprecated*; just write `e == null` instead.

`isPath e`

Return `true` if e evaluates to a path, and `false` otherwise.

`isString e`

Return `true` if e evaluates to a string, and `false` otherwise.

`length e`

Return the length of the list e .

`lessThan e1 e2`

Return `true` if the number $e1$ is less than the number $e2$, and `false` otherwise.

Evaluation aborts if either $e1$ or $e2$ does not evaluate to a number.

`listToAttrs e`

Construct a set from a list specifying the names and values of each attribute. Each element of the list should be a set consisting of a string-valued attribute `name` specifying the name of the attribute, and an attribute `value` specifying its value.

In case of duplicate occurrences of the same name, the first takes precedence.

Example:

```
builtins.listToAttrs
[ { name = "foo"; value = 123; }
  { name = "bar"; value = 456; }
  { name = "bar"; value = 420; }
]
```

evaluates to

```
{ foo = 123; bar = 456; }
```

`map f list`

Apply the function f to each element in the list $list$. For example,

```
map (x: "foo" + x) [ "bar" "bla" "abc" ]
```

evaluates to `["foobar" "foobla" "fooabc"]`.

`mapAttrs f attrset`

Apply function f to every element of $attrset$. For example,

```
builtins.mapAttrs (name: value: value * 10) { a = 1; b = 2; }
```

evaluates to `{ a = 10; b = 20; }`.

`match regex str`

Returns a list if the [extended POSIX regular expression](#) `regex` matches `str` precisely, otherwise returns `null`. Each item in the list is a regex group.

```
builtins.match "ab" "abc"
```

Evaluates to `null`.

```
builtins.match "abc" "abc"
```

Evaluates to `[]`.

```
builtins.match "a(b)(c)" "abc"
```

Evaluates to `["b" "c"]`.

```
builtins.match "[[:space:]]+([[:upper:]]+)[:space:]+\" FOO \"
```

Evaluates to `["FOO"]`.

`mul e1 e2`

Return the product of the numbers `e1` and `e2`.

`outputOf derivation-reference output-name`

Return the output path of a derivation, literally or using a placeholder if needed.

If the derivation has a statically-known output path (i.e. the derivation output is input-addressed, or fixed content-addressed), the output path will just be returned. But if the derivation is content-addressed or if the derivation is itself not-statically produced (i.e. is the output of another derivation), a placeholder will be returned instead.

`derivation reference` must be a string that may contain a regular store path to a derivation, or may be a placeholder reference. If the derivation is produced by a derivation, you must explicitly select `drv.outPath`. This primop can be chained arbitrarily deeply. For instance,

```
builtins.outputOf  
  (builtins.outputOf myDrv "out")  
  "out"
```

will return a placeholder for the output of the output of `myDrv`.

This primop corresponds to the `^` sigil for derivable paths, e.g. as part of installable syntax on the command line.

This function is only available if the [dynamic-derivations](#) experimental feature is enabled.

`parseDrvName s`

Split the string `s` into a package name and version. The package name is everything up to but not including the first dash not followed by a letter, and the version is everything following that dash. The result is returned in a set `{ name, version }`. Thus,

```
builtins.parseDrvName "nix-0.12pre12876" returns { name = "nix"; version =  
  "0.12pre12876"; } .
```

`parseFlakeRef flake-ref`

Parse a flake reference, and return its exploded form.

For example:

```
builtins.parseFlakeRef "github:NixOS/nixpkgs/23.05?dir=lib"
```

evaluates to:

```
{ dir = "lib"; owner = "NixOS"; ref = "23.05"; repo = "nixpkgs"; type =  
  "github"; }
```

This function is only available if the [flakes](#) experimental feature is enabled.

`partition pred list`

Given a predicate function `pred`, this function returns an attrset containing a list named `right`, containing the elements in `list` for which `pred` returned `true`, and a list named `wrong`, containing the elements for which it returned `false`. For example,

```
builtins.partition (x: x > 10) [1 23 9 3 42]
```

evaluates to

```
{ right = [ 23 42 ]; wrong = [ 1 9 3 ]; }
```

path args

An enrichment of the built-in path type, based on the attributes present in *args*. All are optional except `path`:

- `path`

The underlying path.

- `name`

The name of the path when added to the store. This can be used to reference paths that have nix-illegal characters in their names, like `@`.

- `filter`

A function of the type expected by `builtins.filterSource`, with the same semantics.

- `recursive`

When `false`, when `path` is added to the store it is with a flat hash, rather than a hash of the NAR serialization of the file. Thus, `path` must refer to a regular file, not a directory. This allows similar behavior to `fetchurl`. Defaults to `true`.

- `sha256`

When provided, this is the expected hash of the file at the path. Evaluation will fail if the hash is incorrect, and providing a hash allows `builtins.path` to be used even when the `pure-eval` nix config option is on.

pathExists path

Return `true` if the path *path* exists at evaluation time, and `false` otherwise.

placeholder output

Return a placeholder string for the specified *output* that will be substituted by the corresponding output path at build time. Typical outputs would be `"out"`, `"bin"` or `"dev"`.

readDir path

Return the contents of the directory *path* as a set mapping directory entries to the corresponding file type. For instance, if directory `A` contains a regular file `B` and another directory `C`, then `builtins.readDir ./A` will return the set

```
{ B = "regular"; C = "directory"; }
```

The possible values for the file type are "regular", "directory", "symlink" and "unknown".

`readFile path`

Return the contents of the file *path* as a string.

`readFileType p`

Determine the directory entry type of a filesystem node, being one of "directory", "regular", "symlink", or "unknown".

`removeAttrs set list`

Remove the attributes listed in *list* from *set*. The attributes don't have to exist in *set*. For instance,

```
removeAttrs { x = 1; y = 2; z = 3; } [ "a" "x" "z" ]
```

evaluates to `{ y = 2; }`.

`replaceStrings from to s`

Given string *s*, replace every occurrence of the strings in *from* with the corresponding string in *to*.

The argument *to* is lazy, that is, it is only evaluated when its corresponding pattern in *from* is matched in the string *s*

Example:

```
builtins.replaceStrings ["oo" "a"] ["a" "i"] "foobar"
```

evaluates to `"fabir"`.

`seq e1 e2`

Evaluate *e1*, then evaluate and return *e2*. This ensures that a computation is strict in the value of *e1*.

`sort comparator list`

Return *list* in sorted order. It repeatedly calls the function *comparator* with two

elements. The comparator should return `true` if the first element is less than the second, and `false` otherwise. For example,

```
builtins.sort builtins.lessThan [ 483 249 526 147 42 77 ]
```

produces the list `[42 77 147 249 483 526]`.

This is a stable sort: it preserves the relative order of elements deemed equal by the comparator.

`split regex str`

Returns a list composed of non matched strings interleaved with the lists of the [extended POSIX regular expression](#) `regex` matches of `str`. Each item in the lists of matched sequences is a regex group.

```
builtins.split "(a)b" "abc"
```

Evaluates to `["" ["a"] "c"]`.

```
builtins.split "([ac])" "abc"
```

Evaluates to `["" ["a"] "b" ["c"] ""]`.

```
builtins.split "(a)|(c)" "abc"
```

Evaluates to `["" ["a" null] "b" [null "c"] ""]`.

```
builtins.split "[[:upper:]]+" " FOO "
```

Evaluates to `[" " ["FOO"] " "]`.

`splitVersion s`

Split a string representing a version into its components, by the same version splitting logic underlying the version comparison in [`nix-env -u`](#).

`storePath path`

This function allows you to define a dependency on an already existing store path. For example, the derivation attribute `src = builtins.storePath /nix/store/f1d18v1y...-source` causes the derivation to depend on the specified path, which must exist or be substitutable. Note that this differs from a plain path (e.g. `src = /nix/store`

`/f1d18v1y...-source`) in that the latter causes the path to be *copied* again to the Nix store, resulting in a new path (e.g. `/nix/store/ld01dnzc...-source-source`).

Not available in [pure evaluation mode](#).

See also [`builtins.fetchClosure`](#).

`stringLength e`

Return the length of the string `e`. If `e` is not a string, evaluation is aborted.

`sub e1 e2`

Return the difference between the numbers `e1` and `e2`.

`substring start len s`

Return the substring of `s` from character position `start` (zero-based) up to but not including `start + len`. If `start` is greater than the length of the string, an empty string is returned, and if `start + len` lies beyond the end of the string, only the substring up to the end of the string is returned. `start` must be non-negative. For example,

```
builtins.substring 0 3 "nixos"
```

evaluates to `"nix"`.

`tail list`

Return the second to last elements of a list; abort evaluation if the argument isn't a list or is an empty list.

Warning

This function should generally be avoided since it's inefficient: unlike Haskell's `tail`, it takes O(n) time, so recursing over a list by repeatedly calling `tail` takes O(n^2) time.

`throw s`

Throw an error message `s`. This usually aborts Nix expression evaluation, but in `nix-env -qa` and other commands that try to evaluate a set of derivations to get information about those derivations, a derivation that throws an error is silently skipped (which is not the case for `abort`).

toFile name s

Store the string `s` in a file in the Nix store and return its path. The file has suffix `name`. This file can be used as an input to derivations. One application is to write builders “inline”. For instance, the following Nix expression combines the Nix expression for GNU Hello and its build script into one file:

```
{ stdenv, fetchurl, perl }:

stdenv.mkDerivation {
  name = "hello-2.1.1";
  builder = builtins.toFile "builder.sh" "
    source $stdenv/setup

    PATH=$perl/bin:$PATH

    tar xvfz $src
    cd hello-*
    ./configure --prefix=$out
    make
    make install
  ";
  src = fetchurl {
    url = "http://ftp.nluug.nl/pub-gnu/hello/hello-2.1.1.tar.gz";
    sha256 = "1md7jsfd8pa45z73bz1kszpp01yw6x5ljkjk2hx7wl800any6465";
  };
  inherit perl;
}
```

It is even possible for one file to refer to another, e.g.,

```
builder = let
  configFile = builtins.toFile "foo.conf" "
    # This is some dummy configuration file.
    ...
  ";
  in builtins.toFile "builder.sh" "
    source $stdenv/setup
    ...
    cp ${configFile} $out/etc/foo.conf
  ";
```

Note that `${configFile}` is a [string interpolation](#), so the result of the expression `configFile` (i.e., a path like `/nix/store/m7p7jfn445k...-foo.conf`) will be spliced into the resulting string.

It is however *not* allowed to have files mutually referring to each other, like so:

```
let
  foo = builtins.toFile "foo" "...${bar}...";
  bar = builtins.toFile "bar" "...${foo}...";
in foo
```

This is not allowed because it would cause a cyclic dependency in the computation of the cryptographic hashes for `foo` and `bar`.

It is also not possible to reference the result of a derivation. If you are using Nixpkgs, the `writeTextFile` function is able to do that.

`toJSON e`

Return a string containing a JSON representation of `e`. Strings, integers, floats, booleans, nulls and lists are mapped to their JSON equivalents. Sets (except derivations) are represented as objects. Derivations are translated to a JSON string containing the derivation's output path. Paths are copied to the store and represented as a JSON string of the resulting store path.

`toPath s`

DEPRECATED. Use `./. + "/path"` to convert a string into an absolute path. For relative paths, use `./. + "/path"`.

`toString e`

Convert the expression `e` to a string. `e` can be:

- A string (in which case the string is returned unmodified).
- A path (e.g., `toString /foo/bar` yields `"/foo/bar"`).
- A set containing `{ __toString = self: ...; }` or `{ outPath = ...; }`.
- An integer.
- A list, in which case the string representations of its elements are joined with spaces.
- A Boolean (`false` yields `""`, `true` yields `"1"`).
- `null`, which yields the empty string.

`toXML e`

Return a string containing an XML representation of *e*. The main application for `toXML` is to communicate information with the builder in a more structured format than plain environment variables.

Here is an example where this is the case:

```
{ stdenv, fetchurl, libxslt, jira, uberwiki }:

stdenv.mkDerivation (rec {
  name = "web-server";

  buildInputs = [ libxslt ];

  builder = builtins.toFile "builder.sh" "
    source $stdenv/setup
    mkdir $out
    echo "$servlets" | xsltproc ${stylesheet} -> $out/server-conf.xml ①
  ";

  stylesheet = builtins.toFile "stylesheet.xsl" ②
  "<?xml version='1.0' encoding='UTF-8'?>
   <xsl:stylesheet xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
version='1.0'>
    <xsl:template match='/'>
      <Configure>
        <xsl:for-each select='/expr/list/attrs'>
          <Call name='addWebApplication'>
            <Arg><xsl:value-of select=\"$attr[@name = 'path']/string
@value\" /></Arg>
            <Arg><xsl:value-of select=\"$attr[@name = 'war']/path/@value\" /></Arg>
          </Call>
        </xsl:for-each>
      </Configure>
    </xsl:template>
  </xsl:stylesheet>
";
  servlets = builtins.toXML [ ③
    { path = "/bugtracker"; war = jira + "/lib/atlassian-jira.war"; }
    { path = "/wiki"; war = uberwiki + "/uberwiki.war"; }
  ];
})
```

The builder is supposed to generate the configuration file for a [Jetty servlet container](#). A servlet container contains a number of servlets (`*.war` files) each exported under a specific URI prefix. So the servlet configuration is a list of sets containing the `path` and `war` of the servlet (①). This kind of information is difficult to communicate with the normal method of passing information through an environment variable, which just

concatenates everything together into a string (which might just work in this case, but wouldn't work if fields are optional or contain lists themselves). Instead the Nix expression is converted to an XML representation with `toXML`, which is unambiguous and can easily be processed with the appropriate tools. For instance, in the example an XSLT stylesheet (at point ②) is applied to it (at point ①) to generate the XML configuration file for the Jetty server. The XML representation produced at point ③ by `toXML` is as follows:

```
<?xml version='1.0' encoding='utf-8'?>
<expr>
  <list>
    <attrs>
      <attr name="path">
        <string value="/bugtracker" />
      </attr>
      <attr name="war">
        <path value="/nix/store/d1jh9pasa7k2...-jira/lib/atlassian-
jira.war" />
      </attr>
    </attrs>
    <attrs>
      <attr name="path">
        <string value="/wiki" />
      </attr>
      <attr name="war">
        <path value="/nix/store/y6423b1yi4sx...-uberwiki/uberwiki.war" />
      </attr>
    </attrs>
  </list>
</expr>
```

Note that we used the `toFile` built-in to write the builder and the stylesheet “inline” in the Nix expression. The path of the stylesheet is spliced into the builder using the syntax `xsltproc ${stylesheet}`.

`trace e1 e2`

Evaluate `e1` and print its abstract syntax representation on standard error. Then return `e2`. This function is useful for debugging.

`traceVerbose e1 e2`

Evaluate `e1` and print its abstract syntax representation on standard error if `--trace-verbose` is enabled. Then return `e2`. This function is useful for debugging.

`tryEval e`

Try to shallowly evaluate *e*. Return a set containing the attributes `success` (`true` if *e* evaluated successfully, `false` if an error was thrown) and `value`, equalling *e* if successful and `false` otherwise. `tryEval` will only prevent errors created by `throw` or `assert` from being thrown. Errors `tryEval` will not catch are for example those created by `abort` and type errors generated by builtins. Also note that this doesn't evaluate *e* deeply, so `let e = { x = throw ""; }; in (builtins.tryEval e).success` will be `true`. Using `builtins.deepSeq` one can get the expected result: `let e = { x = throw ""; }; in (builtins.tryEval (builtins.deepSeq e e)).success` will be `false`.

`typeOf e`

Return a string representing the type of the value *e*, namely `"int"`, `"bool"`, `"string"`, `"path"`, `"null"`, `"set"`, `"list"`, `"lambda"` or `"float"`.

`zipAttrsWith f list`

Transpose a list of attribute sets into an attribute set of lists, then apply `mapAttrs`.

f receives two arguments: the attribute name and a non-empty list of all values encountered for that attribute name.

The result is an attribute set where the attribute names are the union of the attribute names in each element of `list`. The attribute values are the return values of *f*.

```
builtins.zipAttrsWith
  (name: values: { inherit name values; })
  [ { a = "x"; } { a = "y"; b = "z"; } ]
```

evaluates to

```
{
  a = { name = "a"; values = [ "x" "y" ]; };
  b = { name = "b"; values = [ "z" ]; };
}
```

This section lists advanced topics related to builds and builds performance

Remote Builds

Nix supports remote builds, where a local Nix installation can forward Nix builds to other machines. This allows multiple builds to be performed in parallel and allows Nix to perform multi-platform builds in a semi-transparent way. For instance, if you perform a build for a `x86_64-darwin` on an `i686-linux` machine, Nix can automatically forward the build to a `x86_64-darwin` machine, if available.

To forward a build to a remote machine, it's required that the remote machine is accessible via SSH and that it has Nix installed. You can test whether connecting to the remote Nix instance works, e.g.

```
$ nix store ping --store ssh://mac
```

will try to connect to the machine named `mac`. It is possible to specify an SSH identity file as part of the remote store URI, e.g.

```
$ nix store ping --store ssh://mac?ssh-key=/home/alice/my-key
```

Since builds should be non-interactive, the key should not have a passphrase. Alternatively, you can load identities ahead of time into `ssh-agent` or `gpg-agent`.

If you get the error

```
bash: nix-store: command not found
error: cannot connect to 'mac'
```

then you need to ensure that the `PATH` of non-interactive login shells contains Nix.

Warning

If you are building via the Nix daemon, it is the Nix daemon user account (that is, `root`) that should have SSH access to a user (not necessarily `root`) on the remote machine.

If you can't or don't want to configure `root` to be able to access the remote machine, you can use a private Nix store instead by passing e.g. `--store ~/my-nix` when running a Nix command from the local machine.

The list of remote machines can be specified on the command line or in the Nix configuration file. The former is convenient for testing. For example, the following command

allows you to build a derivation for `x86_64-darwin` on a Linux machine:

```
$ uname
Linux

$ nix build --impure \
  --expr '(with import <nixpkgs> { system = "x86_64-darwin"; }; runCommand
"foo" {} "uname > $out")' \
  --builders 'ssh://mac x86_64-darwin'
[1/0/1 built, 0.0 MiB DL] building foo on ssh://mac

$ cat ./result
Darwin
```

It is possible to specify multiple builders separated by a semicolon or a newline, e.g.

```
--builders 'ssh://mac x86_64-darwin ; ssh://beastie x86_64-freebsd'
```

Each machine specification consists of the following elements, separated by spaces. Only the first element is required. To leave a field at its default, set it to `-`.

1. The URI of the remote store in the format `ssh://[username@]hostname`, e.g. `ssh://nix@mac` or `ssh://mac`. For backward compatibility, `ssh://` may be omitted. The hostname may be an alias defined in your `~/.ssh/config`.
2. A comma-separated list of Nix platform type identifiers, such as `x86_64-darwin`. It is possible for a machine to support multiple platform types, e.g., `i686-linux,x86_64-linux`. If omitted, this defaults to the local platform type.
3. The SSH identity file to be used to log in to the remote machine. If omitted, SSH will use its regular identities.
4. The maximum number of builds that Nix will execute in parallel on the machine. Typically this should be equal to the number of CPU cores. For instance, the machine `itchy` in the example will execute up to 8 builds in parallel.
5. The “speed factor”, indicating the relative speed of the machine. If there are multiple machines of the right type, Nix will prefer the fastest, taking load into account.
6. A comma-separated list of *supported features*. If a derivation has the `requiredSystemFeatures` attribute, then Nix will only perform the derivation on a machine that has the specified features. For instance, the attribute

```
requiredSystemFeatures = [ "kvm" ];
```

will cause the build to be performed on a machine that has the `kvm` feature.

7. A comma-separated list of *mandatory features*. A machine will only be used to build a derivation if all of the machine's mandatory features appear in the derivation's `requiredSystemFeatures` attribute.
8. The (base64-encoded) public host key of the remote machine. If omitted, SSH will use its regular known-hosts file. Specifically, the field is calculated via `base64 -w0 /etc/ssh/ssh_host_ed25519_key.pub`.

For example, the machine specification

```
nix@scratchy.labs.cs.uu.nl  i686-linux      /home/nix/.ssh/id_scratchy_auto
8 1 kvm
nix@itchy.labs.cs.uu.nl     i686-linux      /home/nix/.ssh/id_scratchy_auto
8 2
nix@poochie.labs.cs.uu.nl   i686-linux      /home/nix/.ssh/id_scratchy_auto
1 2 kvm benchmark
```

specifies several machines that can perform `i686-linux` builds. However, `poochie` will only do builds that have the attribute

```
requiredSystemFeatures = [ "benchmark" ];
```

or

```
requiredSystemFeatures = [ "benchmark" "kvm" ];
```

`itchy` cannot do builds that require `kvm`, but `scratchy` does support such builds. For regular builds, `itchy` will be preferred over `scratchy` because it has a higher speed factor.

Remote builders can also be configured in `nix.conf`, e.g.

```
builders = ssh://mac x86_64-darwin ; ssh://beastie x86_64-freebsd
```

Finally, remote builders can be configured in a separate configuration file included in `builders` via the syntax `@file`. For example,

```
builders = @/etc/nix/machines
```

causes the list of machines in `/etc/nix/machines` to be included. (This is the default.)

If you want the builders to use caches, you likely want to set the option `builders-use-`

substitutes in your local `nix.conf`.

To build only on remote builders and disable building on the local machine, you can use the option `--max-jobs 0`.

Tuning Cores and Jobs

Nix has two relevant settings with regards to how your CPU cores will be utilized: `cores` and `max-jobs`. This chapter will talk about what they are, how they interact, and their configuration trade-offs.

- `max-jobs`

Dictates how many separate derivations will be built at the same time. If you set this to zero, the local machine will do no builds. Nix will still substitute from binary caches, and build remotely if remote builders are configured.

- `cores`

Suggests how many cores each derivation should use. Similar to `make -j`.

The `cores` setting determines the value of `NIX_BUILD_CORES`. `NIX_BUILD_CORES` is equal to `cores`, unless `cores` equals `0`, in which case `NIX_BUILD_CORES` will be the total number of cores in the system.

The maximum number of consumed cores is a simple multiplication, `max-jobs * NIX_BUILD_CORES`.

The balance on how to set these two independent variables depends upon each builder's workload and hardware. Here are a few example scenarios on a machine with 24 cores:

<code>max-jobs</code>	<code>cores</code>	<code>NIX_BUILD_CORES</code>	Maximum Processes	Result
1	24	24	24	One derivation will be built at a time, each one can use 24 cores. Undersold if a job can't use 24 cores.
4	6	6	24	Four derivations will be built at once, each given access to six cores.
12	6	6	72	12 derivations will be built at once, each given access to six cores. This configuration is over-

max cores jobs	NIXBUILD_CORES	Maximum Processes	Result
24	1	24	sold. If all 12 derivations being built simultaneously try to use all six cores, the machine's performance will be degraded due to extensive context switching between the 12 builds.
24	0	24	24 derivations can build at the same time, each using a single core. Never oversold, but derivations which require many cores will be very slow to compile.
24	0	576	24 derivations can build at the same time, each using all the available cores of the machine. Very likely to be oversold, and very likely to suffer context switches.

It is up to the derivations' build script to respect host's requested cores-per-build by following the value of the `NIX_BUILD_CORES` environment variable.

Verifying Build Reproducibility

You can use Nix's `diff-hook` setting to compare build results. Note that this hook is only executed if the results differ; it is not used for determining if the results are the same.

For purposes of demonstration, we'll use the following Nix file, `deterministic.nix` for testing:

```
let
  inherit (import <nixpkgs> {}) runCommand;
in {
  stable = runCommand "stable" {} ''
    touch $out
  '';
  unstable = runCommand "unstable" {} ''
    echo $RANDOM > $out
  '';
}
```

Additionally, `nix.conf` contains:

```
diff-hook = /etc/nix/my-diff-hook
run-diff-hook = true
```

where `/etc/nix/my-diff-hook` is an executable file containing:

```
#!/bin/sh
exec >&2
echo "For derivation $3:"
/run/current-system/sw/bin/diff -r "$1" "$2"
```

The diff hook is executed by the same user and group who ran the build. However, the diff hook does not have write access to the store path just built.

Spot-Checking Build Determinism

Verify a path which already exists in the Nix store by passing `--check` to the build command.

If the build passes and is deterministic, Nix will exit with a status code of 0:

```
$ nix-build ./deterministic.nix --attr stable
this derivation will be built:
  /nix/store/z98fasz2jqy9gs0xbvdj939p27jwda38-stable.drv
building '/nix/store/z98fasz2jqy9gs0xbvdj939p27jwda38-stable.drv'...
/nix/store/yyxlzw3vqaas7wfp04g0b1xg51f2czgq-stable

$ nix-build ./deterministic.nix --attr stable --check
checking outputs of '/nix/store/z98fasz2jqy9gs0xbvdj939p27jwda38-stable.drv'...
/nix/store/yyxlzw3vqaas7wfp04g0b1xg51f2czgq-stable
```

If the build is not deterministic, Nix will exit with a status code of 1:

```
$ nix-build ./deterministic.nix --attr unstable
this derivation will be built:
  /nix/store/cgl13lbj1w368r5z8gywipl1ifli7dhk-unstable.drv
building '/nix/store/cgl13lbj1w368r5z8gywipl1ifli7dhk-unstable.drv'...
/nix/store/krpqk0l9ib0ibi1d2w52z293zw455cap-unstable

$ nix-build ./deterministic.nix --attr unstable --check
checking outputs of '/nix/store/cgl13lbj1w368r5z8gywipl1ifli7dhk-
unstable.drv'...
error: derivation '/nix/store/cgl13lbj1w368r5z8gywipl1ifli7dhk-unstable.drv'
may
not be deterministic: output '/nix/store/krpqk0l9ib0ibi1d2w52z293zw455cap-
unstable' differs
```

In the Nix daemon's log, we will now see:

```
For derivation /nix/store/cgl13lbj1w368r5z8gywipl1ifli7dhk-unstable.drv:
1c1
< 8108
---
> 30204
```

Using `--check` with `--keep-failed` will cause Nix to keep the second build's output in a special, `.check` path:

```
$ nix-build ./deterministic.nix --attr unstable --check --keep-failed
checking outputs of '/nix/store/cgl13lbj1w368r5z8gywipl1ifli7dhk-
unstable.drv'...
note: keeping build directory '/tmp/nix-build-unstable.drv-0'
error: derivation '/nix/store/cgl13lbj1w368r5z8gywipl1ifli7dhk-unstable.drv'
may
not be deterministic: output '/nix/store/krpqk0l9ib0ibi1d2w52z293zw455cap-
unstable' differs
from '/nix/store/krpqk0l9ib0ibi1d2w52z293zw455cap-unstable.check'
```

In particular, notice the `/nix/store/krpqk0l9ib0ibi1d2w52z293zw455cap-unstable.check`

output. Nix has copied the build results to that directory where you can examine it.

Note

Check paths are not protected against garbage collection, and this path will be deleted on the next garbage collection.

The path is guaranteed to be alive for the duration of the `diff-hook`'s execution, but may be deleted any time after.

If the comparison is performed as part of automated tooling, please use the `diff-hook` or author your tooling to handle the case where the build was not deterministic and also a check path does not exist.

`--check` is only usable if the derivation has been built on the system already. If the derivation has not been built Nix will fail with the error:

```
error: some outputs of '/nix/store/hzi1h60z2qf0nb85iwnpvrai3j2w7rr6-unstable.drv'  
are not valid, so checking is not possible
```

Run the build without `--check`, and then try with `--check` again.

Using the post-build-hook

Implementation Caveats

Here we use the post-build hook to upload to a binary cache. This is a simple and working example, but it is not suitable for all use cases.

The post build hook program runs after each executed build, and blocks the build loop. The build loop exits if the hook program fails.

Concretely, this implementation will make Nix slow or unusable when the internet is slow or unreliable.

A more advanced implementation might pass the store paths to a user-supplied daemon or queue for processing the store paths outside of the build loop.

Prerequisites

This tutorial assumes you have [configured an S3-compatible binary cache](#), and that the `root` user's default AWS profile can upload to the bucket.

Set up a Signing Key

Use `nix-store --generate-binary-cache-key` to create our public and private signing keys. We will sign paths with the private key, and distribute the public key for verifying the authenticity of the paths.

```
# nix-store --generate-binary-cache-key example-nix-cache-1 /etc/nix  
/key.private /etc/nix/key.public  
# cat /etc/nix/key.public  
example-nix-cache-1:1:cKDz3QCC0mwcztD2eV6Coggp6rqc9DGjWv7C0G+rM=
```

Then update `nix.conf` on any machine that will access the cache. Add the cache URL to `substituters` and the public key to `trusted-public-keys`:

```
substituters = https://cache.nixos.org/ s3://example-nix-cache
trusted-public-keys = cache.nixos.org-
1:6NCHdD59X431o0gWypbMrAURkbJ16ZPMQFGspcDShjY= example-nix-cache-
1:1/cKDz3QCC0mwcztD2eV6Coggp6rqc9DGjWv7C0G+rM=
```

Machines that build for the cache must sign derivations using the private key. On those machines, add the path to the key file to the `secret-key-files` field in their `nix.conf`:

```
secret-key-files = /etc/nix/key.private
```

We will restart the Nix daemon in a later step.

Implementing the build hook

Write the following script to `/etc/nix/upload-to-cache.sh`:

```
#!/bin/sh

set -eu
set -f # disable globbing
export IFS=' '

echo "Uploading paths" $OUT_PATHS
exec nix copy --to "s3://example-nix-cache" $OUT_PATHS
```

Note

The `$OUT_PATHS` variable is a space-separated list of Nix store paths. In this case, we expect and want the shell to perform word splitting to make each output path its own argument to `nix store sign`. Nix guarantees the paths will not contain any spaces, however a store path might contain glob characters. The `set -f` disables globbing in the shell.

Then make sure the hook program is executable by the `root` user:

```
# chmod +x /etc/nix/upload-to-cache.sh
```

Updating Nix Configuration

Edit `/etc/nix/nix.conf` to run our hook, by adding the following configuration snippet at the end:

```
post-build-hook = /etc/nix/upload-to-cache.sh
```

Then, restart the `nix-daemon`.

Testing

Build any derivation, for example:

```
$ nix-build --expr '(import <nixpkgs> {}).writeText "example"  
(builtins.toString builtins.currentTime)'  
this derivation will be built:  
  /nix/store/s4pnfbkalzy5qz57qs6yybna8wylkig6-example.drv  
building '/nix/store/s4pnfbkalzy5qz57qs6yybna8wylkig6-example.drv'...  
running post-build-hook '/home/grahamc/projects/github.com/NixOS/nix/post-  
hook.sh'...  
post-build-hook: Signing paths /nix/store/ibcyipq5gf91838ldx40mjsp0b8w9n18-  
example  
post-build-hook: Uploading paths /nix/store/ibcyipq5gf91838ldx40mjsp0b8w9n18-  
example  
/nix/store/ibcyipq5gf91838ldx40mjsp0b8w9n18-example
```

Then delete the path from the store, and try substituting it from the binary cache:

```
$ rm ./result  
$ nix-store --delete /nix/store/ibcyipq5gf91838ldx40mjsp0b8w9n18-example
```

Now, copy the path back from the cache:

```
$ nix-store --realise /nix/store/ibcyipq5gf91838ldx40mjsp0b8w9n18-example  
copying path '/nix/store/m8bmqrch6l3h8s0k3d673xpmipcdpsa-example' from  
's3://example-nix-cache'...  
warning: you did not specify '--add-root'; the result might be removed by the  
garbage collector  
/nix/store/m8bmqrch6l3h8s0k3d673xpmipcdpsa-example
```

Conclusion

We now have a Nix installation configured to automatically sign and upload every local build

to a remote binary cache.

Before deploying this to production, be sure to consider the [implementation caveats](#).

This section lists commands and options that you can use when you work with Nix.

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose / -v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- `--quiet`

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- `--log-format` *format*

This option can be used to change the output of the log format, with *format* being one of:

- `raw`

This is the raw format, as outputted by nix-build.

- `internal-json`

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- `--no-build-output` / `-Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- `--max-jobs` / `-j` *number*

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If

the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- `--readonly-mode`

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- `--arg name value`

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a `default value` (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.
  system ? builtins.currentSystem
  ...
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgname`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgname --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- `--argstr name value`

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- `--attr / -A attrPath`

Select an attribute from the top-level Nix expression being evaluated. (`nix-env` , `nix-instantiate` , `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression `e`, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- `--expr / -E`

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate` , `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- `NIX_IGNORE_SYMLINK_STORE`

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- [NIX_STORE_DIR](#)

Overrides the location of the Nix store (default `prefix/store`).

- [NIX_DATA_DIR](#)

Overrides the location of the Nix static data directory (default `prefix/share`).

- [NIX_LOG_DIR](#)

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- [NIX_STATE_DIR](#)

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- [NIX_CONF_DIR](#)

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- [NIX_CONFIG](#)

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- [NIX_USER_CONF_FILES](#)

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [XDG_CONFIG_HOME](#) (default `~/.config`)
- [XDG_STATE_HOME](#) (default `~/.local/state`)
- [XDG_CACHE_HOME](#) (default `~/.cache`)

Main Commands

This section lists commands and options that you can use when you work with Nix.

Name

`nix-build` - build a Nix expression

Synopsis

```
nix-build [paths...] [--arg name value] [--argstr name value] [{ --attr | -A } attrPath]
[ --no-out-link ] [ --dry-run ] [{ --out-link | -o } outlink]
```

Disambiguation

This man page describes the command `nix-build`, which is distinct from `nix build`. For documentation on the latter, run `nix build --help` or see `man nix3-build`.

Description

The `nix-build` command builds the derivations described by the Nix expressions in *paths*. If the build succeeds, it places a symlink to the result in the current directory. The symlink is called `result`. If there are multiple Nix expressions, or the Nix expressions evaluate to multiple derivations, multiple sequentially numbered symlinks are created (`result`, `result-2`, and so on).

If no *paths* are specified, then `nix-build` will use `default.nix` in the current directory, if it exists.

If an element of *paths* starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must include a single top-level directory containing at least a file named `default.nix`.

`nix-build` is essentially a wrapper around `nix-instantiate` (to translate a high-level Nix expression to a low-level `store derivation`) and `nix-store --realise` (to build the store derivation).

Warning

The result of the build is automatically registered as a root of the Nix garbage collector. This root disappears automatically when the `result` symlink is deleted or renamed. So don't rename the symlink.

Options

All options not listed here are passed to `nix-store --realise`, except for `--arg` and `--attr` / `-A` which are passed to `nix-instantiate`.

- `--no-out-link`

Do not create a symlink to the output path. Note that as a result the output does not become a root of the garbage collector, and so might be deleted by `nix-store --gc`.

- `--dry-run`

Show what store paths would be built or downloaded.

- `--out-link` / `-o outlink`

Change the name of the symlink to the output path created from `result` to `outlink`.

Special exit codes for build failure

1xx status codes are used when requested builds failed. The following codes are in use:

- `100` Generic build failure

The builder process returned with a non-zero exit code.

- `101` Build timeout

The build was aborted because it did not complete within the specified `timeout`.

- `102` Hash mismatch

The build output was rejected because it does not match the `outputHash` attribute of `the derivation`.

- `104` Not deterministic

The build succeeded in check mode but the resulting output is not binary reproducible.

With the `--keep-going` flag it's possible for multiple failures to occur. In this case the 1xx status codes are or combined using [bitwise OR](#).

```
0b1100100
 ^^^^
 |||`- timeout
 ||`-- output hash mismatch
 |`--- build failure
 `---- not deterministic
```

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose / -v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- `raw`

This is the raw format, as outputted by nix-build.

- `internal-json`

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- `--no-build-output / -Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix /var/log/nix`.

- `--max-jobs / -j number`

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- `--cores`

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in `Nixpkgs`, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- `--max-silent-time`

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- `--timeout`

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- `--keep-going / -k`

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.
  system ? builtins.currentSystem
  ...
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgname`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgname --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- `--argstr name value`

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- `--attr / -A attrPath`

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression *e*, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- `--expr / -E`

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- [--repair](#)

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under [nix-store --repair-path](#).

Common Environment Variables

Most Nix commands interpret the following environment variables:

- [IN_NIX_SHELL](#)

Indicator that tells if the current environment was set up by [nix-shell](#). It can have the values [pure](#) or [impure](#).

- [NIX_PATH](#)

A colon-separated list of directories used to look up the location of Nix expressions using [paths](#) enclosed in angle brackets (i.e., [`<path>`](#)), e.g. [`/home/eelco/Dev:/etc/nixos`](#). It can be extended using the [-I option](#).

If [NIX_PATH](#) is not set at all, Nix will fall back to the following list in [impure](#) and [unrestricted](#) evaluation mode:

1. [`\$HOME/.nix-defexpr/channels`](#)
2. [`nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`](#)
3. [`/nix/var/nix/profiles/per-user/root/channels`](#)

If [NIX_PATH](#) is set to an empty string, resolving search paths will always fail. For example, attempting to use [`<nixpkgs>`](#) will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- [NIX_IGNORE_SYMLINK_STORE](#)

Normally, the Nix store directory (typically [`/nix/store`](#)) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with [`/nix/store`](#) resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where [`/nix/store`](#) resolves differently. If you are sure that you’re not going to do that, you can set [NIX_IGNORE_SYMLINK_STORE](#) to [1](#).

Note that if you're symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you're better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- **TMPDIR**

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- **NIX_REMOTE**

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [`NIX_SHOW_STATS`](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [`NIX_COUNT_CALLS`](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [`GC_INITIAL_HEAP_SIZE`](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [`XDG_CONFIG_HOME`](#) (default `~/.config`)
- [`XDG_STATE_HOME`](#) (default `~/.local/state`)
- [`XDG_CACHE_HOME`](#) (default `~/.cache`)

Examples

```
$ nix-build '<nixpkgs>' --attr firefox
store derivation is /nix/store/qybprl8sz2lc...-firefox-1.5.0.7.drv
/nix/store/d18hyl92g30l...-firefox-1.5.0.7

$ ls -l result
lrwxrwxrwx ... result -> /nix/store/d18hyl92g30l...-firefox-1.5.0.7

$ ls ./result/bin/
firefox  firefox-config
```

If a derivation has multiple outputs, `nix-build` will build the default (first) output. You can also build all outputs:

```
$ nix-build '<nixpkgs>' --attr openssl.all
```

This will create a symlink for each output named `result-outputname`. The suffix is omitted if the output name is `out`. So if `openssl` has outputs `out`, `bin` and `man`, `nix-build` will create symlinks `result`, `result-bin` and `result-man`. It's also possible to build a specific output:

```
$ nix-build '<nixpkgs>' --attr openssl.man
```

This will create a symlink `result-man`.

Build a Nix expression given on the command line:

```
$ nix-build --expr 'with import <nixpkgs> { }; runCommand "foo" { } "echo bar > $out"'
$ cat ./result
bar
```

Build the GNU Hello package from the latest revision of the master branch of Nixpkgs:

```
$ nix-build https://github.com/NixOS/nixpkgs/archive/master.tar.gz --attr hello
```

Name

`nix-shell` - start an interactive shell based on a Nix expression

Synopsis

```
nix-shell [ --arg name value] [ --argstr name value] [{ --attr | -A } attrPath]
[ --command cmd] [ --run cmd] [ --exclude regexp] [ --pure ][ --keep name] {{ --packages
| -p } {packages | expressions} ... | [path]}
```

Disambiguation

This man page describes the command `nix-shell`, which is distinct from `nix shell`. For documentation on the latter, run `nix shell --help` or see `man nix3-shell`.

Description

The command `nix-shell` will build the dependencies of the specified derivation, but not the derivation itself. It will then start an interactive shell in which all environment variables defined by the derivation *path* have been set to their corresponding values, and the script `$stdenv/setup` has been sourced. This is useful for reproducing the environment of a derivation for development.

If *path* is not given, `nix-shell` defaults to `shell.nix` if it exists, and `default.nix` otherwise.

If *path* starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must include a single top-level directory containing at least a file named `default.nix`.

If the derivation defines the variable `shellHook`, it will be run after `$stdenv/setup` has been sourced. Since this hook is not executed by regular Nix builds, it allows you to perform initialisation specific to `nix-shell`. For example, the derivation attribute

```
shellHook =  
''  
  echo "Hello shell"  
  export SOME_API_TOKEN=$(cat ~/.config/some-app/api-token)"  
'';
```

will cause `nix-shell` to print `Hello shell` and set the `SOME_API_TOKEN` environment variable to a user-configured value.

Options

All options not listed here are passed to `nix-store --realise`, except for `--arg` and `--attr / -A` which are passed to `nix-instantiate`.

- `--command cmd`

In the environment of the derivation, run the shell command `cmd`. This command is executed in an interactive shell. (Use `--run` to use a non-interactive shell instead.) However, a call to `exit` is implicitly added to the command, so the shell will exit after running the command. To prevent this, add `return` at the end; e.g. `--command "echo Hello; return"` will print `Hello` and then drop you into the interactive shell. This can be useful for doing any additional initialisation.

- `--run cmd`

Like `--command`, but executes the command in a non-interactive shell. This means (among other things) that if you hit Ctrl-C while the command is running, the shell exits.

- `--exclude regexp`

Do not build any dependencies whose store path matches the regular expression `regexp`. This option may be specified multiple times.

- `--pure`

If this flag is specified, the environment is almost entirely cleared before the interactive shell is started, so you get an environment that more closely corresponds to the “real” Nix build. A few variables, in particular `HOME`, `USER` and `DISPLAY`, are retained.

- `--packages / -p packages...`

Set up an environment in which the specified packages are present. The command line arguments are interpreted as attribute names inside the Nix Packages collection. Thus, `nix-shell --packages libjpeg openjdk` will start a shell in which the packages denoted by the attribute names `libjpeg` and `openjdk` are present.

- **-i interpreter**

The chained script interpreter to be invoked by `nix-shell`. Only applicable in `#! -` scripts (described below).

- **--keep name**

When a `--pure` shell is started, keep the listed environment variables.

Common Options

Most Nix commands accept the following command-line options:

- **--help**

Prints out a summary of the command syntax and exits.

- **--version**

Prints out the Nix version number on standard output and exits.

- **--verbose / -v**

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- **0 “Errors only”**

Only print messages explaining why the Nix invocation failed.

- **1 “Informational”**

Print *useful* messages about what Nix is doing. This is the default.

- **2 “Talkative”**

Print more informational messages.

- **3 “Chatty”**

Print even more informational messages.

- **4 “Debug”**

Print debug information.

- `5` “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- `raw`

This is the raw format, as outputted by nix-build.

- `internal-json`

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output** / `-Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix`

`/var/log/nix .`

- `--max-jobs / -j number`

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- `--cores`

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- `--max-silent-time`

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- `--timeout`

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- `--keep-going / -k`

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- `--keep-failed / -K`

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg** *name value*

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.
  system ? builtins.currentSystem
  ...
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgnome --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- `--argstr name value`

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- `--attr / -A attrPath`

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression `e`, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- `--expr / -E`

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store`

--repair-path .

Environment variables

- `NIX_BUILD_SHELL`

Shell used to start the interactive environment. Defaults to the `bash` found in `<nixpkgs>`, falling back to the `bash` found in `PATH` if not found.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- `NIX_IGNORE_SYMLINK_STORE`

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different

results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you're not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you're symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you're better off using `bind` mount points, e.g.,

```
$ mkdir /nix  
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **`NIX_STORE_DIR`**

Overrides the location of the Nix store (default `prefix/store`).

- **`NIX_DATA_DIR`**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **`NIX_LOG_DIR`**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **`NIX_STATE_DIR`**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **`NIX_CONF_DIR`**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **`NIX_CONFIG`**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **`NIX_USER_CONF_FILES`**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- **`TMPDIR`**

Use the specified directory to store temporary files. In particular, this includes

temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- **NIX_REMOTE**

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- **NIX_SHOW_STATS**

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- **NIX_COUNT_CALLS**

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- **GC_INITIAL_HEAP_SIZE**

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- `XDG_CONFIG_HOME` (default `~/.config`)
- `XDG_STATE_HOME` (default `~/.local/state`)
- `XDG_CACHE_HOME` (default `~/.cache`)

Examples

To build the dependencies of the package Pan, and start an interactive shell in which to build it:

```
$ nix-shell '<nixpkgs>' --attr pan
[nix-shell]$ eval ${unpackPhase:-unpackPhase}
[nix-shell]$ cd $sourceRoot
[nix-shell]$ eval ${patchPhase:-patchPhase}
[nix-shell]$ eval ${configurePhase:-configurePhase}
[nix-shell]$ eval ${buildPhase:-buildPhase}
[nix-shell]$ ./pan/gui/pan
```

The reason we use form `eval ${configurePhase:-configurePhase}` here is because those packages that override these phases do so by exporting the overridden values in the environment variable of the same name. Here bash is being told to either evaluate the contents of 'configurePhase', if it exists as a variable, otherwise evaluate the `configurePhase` function.

To clear the environment first, and do some additional automatic initialisation of the interactive shell:

```
$ nix-shell '<nixpkgs>' --attr pan --pure \
  --command 'export NIX_DEBUG=1; export NIX_CORES=8; return'
```

Nix expressions can also be given on the command line using the `-E` and `-p` flags. For instance, the following starts a shell containing the packages `sqlite` and `libX11`:

```
$ nix-shell --expr 'with import <nixpkgs> { }; runCommand "dummy" { buildInputs
  = [ sqlite xorg.libX11 ]; } """'
```

A shorter way to do the same is:

```
$ nix-shell --packages sqlite xorg.libX11
[nix-shell]$ echo $NIX_LDFLAGS
... -L/nix/store/j1zg5v...-sqlite-3.8.0.2/lib -L/nix/store/0gmcz9...-libX11-1.6.1/lib
...
...
```

Note that `-p` accepts multiple full nix expressions that are valid in the `buildInputs = [...]` shown above, not only package names. So the following is also legal:

```
$ nix-shell --packages sqlite 'git.override { withManual = false; }'
```

The `-p` flag looks up Nixpkgs in the Nix search path. You can override it by passing `-I` or setting `NIX_PATH`. For example, the following gives you a shell containing the Pan package

from a specific revision of Nixpkgs:

```
$ nix-shell --packages pan -I nixpkgs=https://github.com/NixOS/nixpkgs/archive/8a3eea054838b55aca962c3fbde9c83c102b8bf2.tar.gz
```

```
[nix-shell:~]$ pan --version
Pan 0.139
```

Use as a #!-interpreter

You can use `nix-shell` as a script interpreter to allow scripts written in arbitrary languages to obtain their own dependencies via Nix. This is done by starting the script with the following lines:

```
#! /usr/bin/env nix-shell
#! nix-shell -i real-interpreter --packages packages
```

where *real-interpreter* is the “real” script interpreter that will be invoked by `nix-shell` after it has obtained the dependencies and initialised the environment, and *packages* are the attribute names of the dependencies in Nixpkgs.

The lines starting with `#! nix-shell` specify `nix-shell` options (see above). Note that you cannot write `#! /usr/bin/env nix-shell -i ...` because many operating systems only allow one argument in `#!` lines.

For example, here is a Python script that depends on Python and the `prettytable` package:

```
#! /usr/bin/env nix-shell
#! nix-shell -i python --packages python pythonPackages.prettytable

import prettytable

# Print a simple table.
t = prettytable.PrettyTable(["N", "N^2"])
for n in range(1, 10): t.add_row([n, n * n])
print t
```

Similarly, the following is a Perl script that specifies that it requires Perl and the `HTML::TokeParser::Simple` and `LWP` packages:

```
#!/usr/bin/env nix-shell
#! nix-shell -i perl --packages perl perlPackages.HTMLTokeParserSimple
perlPackages.LWP

use HTML::TokeParser::Simple;

# Fetch nixos.org and print all hrefs.
my $p = HTML::TokeParser::Simple->new(url => 'http://nixos.org/');

while (my $token = $p->get_tag("a")) {
    my $href = $token->get_attr("href");
    print "$href\n" if $href;
}
```

Sometimes you need to pass a simple Nix expression to customize a package like Terraform:

```
#!/usr/bin/env nix-shell
#! nix-shell -i bash --packages "terraform.withPlugins (plugins: [
plugins.openstack ])"
```

terraform apply

Note

You must use double quotes (") when passing a simple Nix expression in a nix-shell shebang.

Finally, using the merging of multiple nix-shell shebangs the following Haskell script uses a specific branch of Nixpkgs/NixOS (the 20.03 stable branch):

```
#! /usr/bin/env nix-shell
#! nix-shell -i runghc --packages "haskellPackages.ghcWithPackages (ps:
[ps.download-curl ps.tagSoup])"
#! nix-shell -I nixpkgs=https://github.com/NixOS/nixpkgs/archive/nixos-
20.03.tar.gz

import Network.Curl.Download
import Text.HTML.TagSoup
import Data.Either
import Data.ByteString.Char8 (unpack)

-- Fetch nixos.org and print all hrefs.
main = do
  resp <- openURI "https://nixos.org/"
  let tags = filter (isTagOpenName "a") $ parseTags $ unpack $ fromRight
  undefined resp
  let tags' = map (fromAttrib "href") tags
  mapM_ putStrLn $ filter (/= "") tags'
```

If you want to be even more precise, you can specify a specific revision of Nixpkgs:

```
#! nix-shell -I nixpkgs=https://github.com/NixOS/nixpkgs/archive
/0672315759b3e15e2121365f067c1c8c56bb4722.tar.gz
```

The examples above all used `-p` to get dependencies from Nixpkgs. You can also use a Nix expression to build your own dependencies. For example, the Python example could have been written as:

```
#! /usr/bin/env nix-shell
#! nix-shell deps.nix -i python
```

where the file `deps.nix` in the same directory as the `#!` -script contains:

```
with import <nixpkgs> {};
runCommand "dummy" { buildInputs = [ python pythonPackages.prettytable ]; } ""
```

Name

`nix-store` - manipulate or query the Nix store

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`
- `--add`
- `--add-fixed`
- `--verify`
- `--verify-path`
- `--repair-path`
- `--dump`
- `--restore`
- `--export`
- `--import`
- `--optimise`
- `--read-log`
- `--dump-db`
- `--load-db`
- `--print-env`
- `--generate-binary-cache-key`

These pages can be viewed offline:

- `man nix-store-<operation>`.

Example: `man nix-store-realise`

- `nix-store --help --<operation>`

Example: `nix-store --help --realise`

Name

`nix-store --add-fixed` - add paths to store using given hashing algorithm

Synopsis

`nix-store --add-fixed [--recursive] algorithm paths...`

Description

The operation `--add-fixed` adds the specified paths to the Nix store. Unlike `--add` paths are registered using the specified hashing algorithm, resulting in the same output path as a fixed-output derivation. This can be used for sources that are not available from a public url or broke since the download expression was written.

This operation has the following options:

- `--recursive`

Use recursive instead of flat hashing mode, used when adding directories to the store.

Options

The following options are allowed for all `nix-store` operations, but may not always have an effect.

- `--add-root` *path*

Causes the result of a realisation (`--realise` and `--force-realise`) to be registered as a root of the garbage collector. *path* will be created as a symlink to the resulting store path. In addition, a uniquely named symlink to *path* will be created in `/nix/var/nix/gcroots/auto/`. For instance,

```
$ nix-store --add-root /home/eelco/bla/result --realise ...  
  
$ ls -l /nix/var/nix/gcroots/auto  
lrwxrwxrwx    1 ... 2005-03-13 21:10 dn54lcypm8f8... -> /home/eelco  
/bla/result  
  
$ ls -l /home/eelco/bla/result  
lrwxrwxrwx    1 ... 2005-03-13 21:10 /home/eelco/bla/result -> /nix/store  
/1r11343n6qd4...-f-spot-0.0.10
```

Thus, when `/home/eelco/bla/result` is removed, the GC root in the `auto` directory becomes a dangling symlink and will be ignored by the collector.

Warning

Note that it is not possible to move or rename GC roots, since the symlink in the `auto` directory will still point to the old location.

If there are multiple results, then multiple symlinks will be created by sequentially numbering symlinks beyond the first one (e.g., `foo`, `foo-2`, `foo-3`, and so on).

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose` / `-v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- **0** “Errors only”

Only print messages explaining why the Nix invocation failed.

- **1** “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- **2** “Talkative”

Print more informational messages.

- **3** “Chatty”

Print even more informational messages.

- **4** “Debug”

Print debug information.

- **5** “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to **-v** / **--verbose**.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- **raw**

This is the raw format, as outputted by nix-build.

- **internal-json**

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output / -Q**

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a `default value` (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.  
  system ? builtins.currentSystem  
  ...  
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgnome --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- **--argstr** *name value*

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- **--attr** / **-A** *attrPath*

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression *e*, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- **--expr** / **-E**

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the

built packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under [nix-store --repair-path](#).

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **NIX_IGNORE_SYMLINK_STORE**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [XDG_CONFIG_HOME](#) (default `~/.config`)
- [XDG_STATE_HOME](#) (default `~/.local/state`)

- `XDG_CACHE_HOME` (default `~/.cache`)

Example

```
$ nix-store --add-fixed sha256 ./hello-2.10.tar.gz  
/nix/store/3x7dwzq014bbblazs7kq20p9hyzz0qh8g-hello-2.10.tar.gz
```

Name

`nix-store --add` - add paths to Nix store

Synopsis

`nix-store --add paths...`

Description

The operation `--add` adds the specified paths to the Nix store. It prints the resulting paths in the Nix store on standard output.

Options

The following options are allowed for all `nix-store` operations, but may not always have an effect.

- `--add-root path`

Causes the result of a realisation (`--realise` and `--force-realise`) to be registered as a root of the garbage collector. *path* will be created as a symlink to the resulting store path. In addition, a uniquely named symlink to *path* will be created in `/nix/var/nix/gcroots/auto/`. For instance,

```
$ nix-store --add-root /home/eelco/bla/result --realise ...
```

```
$ ls -l /nix/var/nix/gcroots/auto
lrwxrwxrwx    1 ... 2005-03-13 21:10 dn54lcypm8f8... -> /home/eelco
/bla/result
```

```
$ ls -l /home/eelco/bla/result
lrwxrwxrwx    1 ... 2005-03-13 21:10 /home/eelco/bla/result -> /nix/store
/1r11343n6qd4...-f-spot-0.0.10
```

Thus, when `/home/eelco/bla/result` is removed, the GC root in the `auto` directory becomes a dangling symlink and will be ignored by the collector.

Warning

Note that it is not possible to move or rename GC roots, since the symlink in the `auto` directory will still point to the old location.

If there are multiple results, then multiple symlinks will be created by sequentially numbering symlinks beyond the first one (e.g., `foo`, `foo-2`, `foo-3`, and so on).

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose / -v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- `0` “Errors only”

Only print messages explaining why the Nix invocation failed.

- `1` “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- `2` “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- `raw`

This is the raw format, as outputted by nix-build.

- `internal-json`

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- `--no-build-output / -Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix /var/log/nix`.

- `--max-jobs / -j number`

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- `--cores`

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in `Nixpkgs`, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- `--max-silent-time`

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- `--timeout`

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- `--keep-going / -k`

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.
  system ? builtins.currentSystem
  ...
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgname`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgname --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- `--argstr name value`

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- `--attr / -A attrPath`

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression *e*, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- `--expr / -E`

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- [--repair](#)

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under [nix-store --repair-path](#).

Common Environment Variables

Most Nix commands interpret the following environment variables:

- [IN_NIX_SHELL](#)

Indicator that tells if the current environment was set up by [nix-shell](#). It can have the values [pure](#) or [impure](#).

- [NIX_PATH](#)

A colon-separated list of directories used to look up the location of Nix expressions using [paths](#) enclosed in angle brackets (i.e., [`<path>`](#)), e.g. [`/home/eelco/Dev:/etc/nixos`](#). It can be extended using the [-I option](#).

If [NIX_PATH](#) is not set at all, Nix will fall back to the following list in [impure](#) and [unrestricted](#) evaluation mode:

1. [`\$HOME/.nix-defexpr/channels`](#)
2. [`nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`](#)
3. [`/nix/var/nix/profiles/per-user/root/channels`](#)

If [NIX_PATH](#) is set to an empty string, resolving search paths will always fail. For example, attempting to use [`<nixpkgs>`](#) will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- [NIX_IGNORE_SYMLINK_STORE](#)

Normally, the Nix store directory (typically [`/nix/store`](#)) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with [`/nix/store`](#) resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where [`/nix/store`](#) resolves differently. If you are sure that you’re not going to do that, you can set [NIX_IGNORE_SYMLINK_STORE](#) to [1](#).

Note that if you're symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you're better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- **TMPDIR**

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- **NIX_REMOTE**

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [`NIX_SHOW_STATS`](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [`NIX_COUNT_CALLS`](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [`GC_INITIAL_HEAP_SIZE`](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [`XDG_CONFIG_HOME`](#) (default `~/.config`)
- [`XDG_STATE_HOME`](#) (default `~/.local/state`)
- [`XDG_CACHE_HOME`](#) (default `~/.cache`)

Example

```
$ nix-store --add ./foo.c  
/nix/store/m7lrha58ph6rcnv109yzx1nk1cj7k7zf-foo.c
```

Name

`nix-store --delete` - delete store paths

Synopsis

`nix-store --delete [--ignore-liveness] paths...`

Description

The operation `--delete` deletes the store paths *paths* from the Nix store, but only if it is safe to do so; that is, when the path is not reachable from a root of the garbage collector. This means that you can only delete paths that would also be deleted by `nix-store --gc`. Thus, `--delete` is a more targeted version of `--gc`.

With the option `--ignore-liveness`, reachability from the roots is ignored. However, the path still won't be deleted if there are other paths in the store that refer to it (i.e., depend on it).

Options

The following options are allowed for all `nix-store` operations, but may not always have an effect.

- `--add-root` *path*

Causes the result of a realisation (`--realise` and `--force-realise`) to be registered as a root of the garbage collector. *path* will be created as a symlink to the resulting store path. In addition, a uniquely named symlink to *path* will be created in `/nix/var/nix/gcroots/auto/`. For instance,

```
$ nix-store --add-root /home/eelco/bla/result --realise ...  
  
$ ls -l /nix/var/nix/gcroots/auto  
lrwxrwxrwx 1 ... 2005-03-13 21:10 dn54lcypm8f8... -> /home/eelco  
/bla/result  
  
$ ls -l /home/eelco/bla/result  
lrwxrwxrwx 1 ... 2005-03-13 21:10 /home/eelco/bla/result -> /nix/store  
/1r11343n6qd4...-f-spot-0.0.10
```

Thus, when `/home/eelco/bla/result` is removed, the GC root in the `auto` directory becomes a dangling symlink and will be ignored by the collector.

Warning

Note that it is not possible to move or rename GC roots, since the symlink in the `auto` directory will still point to the old location.

If there are multiple results, then multiple symlinks will be created by sequentially numbering symlinks beyond the first one (e.g., `foo`, `foo-2`, `foo-3`, and so on).

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose` / `-v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- **0** “Errors only”

Only print messages explaining why the Nix invocation failed.

- **1** “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- **2** “Talkative”

Print more informational messages.

- **3** “Chatty”

Print even more informational messages.

- **4** “Debug”

Print debug information.

- **5** “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to **-v** / **--verbose**.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- **raw**

This is the raw format, as outputted by nix-build.

- **internal-json**

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output / -Q**

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a `default value` (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.  
  system ? builtins.currentSystem  
  ...  
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgnome --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- **--argstr** *name value*

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- **--attr** / **-A** *attrPath*

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression *e*, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- **--expr** / **-E**

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the

built packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under [nix-store --repair-path](#).

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **NIX_IGNORE_SYMLINK_STORE**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [XDG_CONFIG_HOME](#) (default `~/.config`)
- [XDG_STATE_HOME](#) (default `~/.local/state`)

- `XDG_CACHE_HOME` (default `~/cache`)

Example

```
$ nix-store --delete /nix/store/zq0h41l75v1b4z45kzgjjmsjxvcv1qk7-mesa-6.4
0 bytes freed (0.00 MiB)
error: cannot delete path `/nix/store/zq0h41l75v1b4z45kzgjjmsjxvcv1qk7-
mesa-6.4' since it is still alive
```

Name

`nix-store --dump-db` - export Nix database

Synopsis

`nix-store --dump-db [paths...]`

Description

The operation `--dump-db` writes a dump of the Nix database to standard output. It can be loaded into an empty Nix store using `--load-db`. This is useful for making backups and when migrating to different database schemas.

By default, `--dump-db` will dump the entire Nix database. When one or more store paths are passed, only the subset of the Nix database for those store paths is dumped. As with `--export`, the user is responsible for passing all the store paths for a closure. See `--export` for an example.

Options

The following options are allowed for all `nix-store` operations, but may not always have an effect.

- `--add-root path`

Causes the result of a realisation (`--realise` and `--force-realise`) to be registered as a root of the garbage collector. `path` will be created as a symlink to the resulting store path. In addition, a uniquely named symlink to `path` will be created in `/nix/var/nix/gcroots/auto/`. For instance,

```
$ nix-store --add-root /home/eelco/bla/result --realise ...  
  
$ ls -l /nix/var/nix/gcroots/auto  
lrwxrwxrwx    1 ... 2005-03-13 21:10 dn54lcypm8f8... -> /home/eelco  
/bla/result  
  
$ ls -l /home/eelco/bla/result  
lrwxrwxrwx    1 ... 2005-03-13 21:10 /home/eelco/bla/result -> /nix/store  
/1r11343n6qd4...-f-spot-0.0.10
```

Thus, when `/home/eelco/bla/result` is removed, the GC root in the `auto` directory becomes a dangling symlink and will be ignored by the collector.

Warning

Note that it is not possible to move or rename GC roots, since the symlink in the `auto` directory will still point to the old location.

If there are multiple results, then multiple symlinks will be created by sequentially numbering symlinks beyond the first one (e.g., `foo`, `foo-2`, `foo-3`, and so on).

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose` / `-v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- **0** “Errors only”

Only print messages explaining why the Nix invocation failed.

- **1** “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- **2** “Talkative”

Print more informational messages.

- **3** “Chatty”

Print even more informational messages.

- **4** “Debug”

Print debug information.

- **5** “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to **-v** / **--verbose**.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- **raw**

This is the raw format, as outputted by nix-build.

- **internal-json**

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output / -Q**

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a `default value` (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.  
  system ? builtins.currentSystem  
  ...  
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgnome --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- **--argstr** *name value*

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- **--attr** / **-A** *attrPath*

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression *e*, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- **--expr** / **-E**

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the

built packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under [nix-store --repair-path](#).

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **NIX_IGNORE_SYMLINK_STORE**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [XDG_CONFIG_HOME](#) (default `~/.config`)
- [XDG_STATE_HOME](#) (default `~/.local/state`)

- `XDG_CACHE_HOME` (default `~/.cache`)

Name

`nix-store --dump` - write a single path to a Nix Archive

Synopsis

`nix-store --dump path`

Description

The operation `--dump` produces a NAR (Nix ARchive) file containing the contents of the file system tree rooted at *path*. The archive is written to standard output.

A NAR archive is like a TAR or Zip archive, but it contains only the information that Nix considers important. For instance, timestamps are elided because all files in the Nix store have their timestamp set to 0 anyway. Likewise, all permissions are left out except for the execute bit, because all files in the Nix store have 444 or 555 permission.

Also, a NAR archive is *canonical*, meaning that “equal” paths always produce the same NAR archive. For instance, directory entries are always sorted so that the actual on-disk order doesn’t influence the result. This means that the cryptographic hash of a NAR dump of a path is usable as a fingerprint of the contents of the path. Indeed, the hashes of store paths stored in Nix’s database (see `nix-store --query --hash`) are SHA-256 hashes of the NAR dump of each store path.

NAR archives support filenames of unlimited length and 64-bit file sizes. They can contain regular files, directories, and symbolic links, but not other types of files (such as device nodes).

A Nix archive can be unpacked using `nix-store --restore`.

Options

The following options are allowed for all `nix-store` operations, but may not always have an effect.

- `--add-root path`

Causes the result of a realisation (`--realise` and `--force-realise`) to be registered

as a root of the garbage collector. *path* will be created as a symlink to the resulting store path. In addition, a uniquely named symlink to *path* will be created in `/nix/var/nix/gcroots/auto/`. For instance,

```
$ nix-store --add-root /home/eelco/bla/result --realise ...  
  
$ ls -l /nix/var/nix/gcroots/auto  
lrwxrwxrwx 1 ... 2005-03-13 21:10 dn54lcypm8f8... -> /home/eelco  
/bla/result  
  
$ ls -l /home/eelco/bla/result  
lrwxrwxrwx 1 ... 2005-03-13 21:10 /home/eelco/bla/result -> /nix/store  
/1r11343n6qd4...-f-spot-0.0.10
```

Thus, when `/home/eelco/bla/result` is removed, the GC root in the `auto` directory becomes a dangling symlink and will be ignored by the collector.

Warning

Note that it is not possible to move or rename GC roots, since the symlink in the `auto` directory will still point to the old location.

If there are multiple results, then multiple symlinks will be created by sequentially numbering symlinks beyond the first one (e.g., `foo`, `foo-2`, `foo-3`, and so on).

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose` / `-v`

Increases the level of verbosity of diagnostic messages printed on standard error. For

each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- **0** “Errors only”

Only print messages explaining why the Nix invocation failed.

- **1** “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- **2** “Talkative”

Print more informational messages.

- **3** “Chatty”

Print even more informational messages.

- **4** “Debug”

Print debug information.

- **5** “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to **-v** / **--verbose**.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- **raw**

This is the raw format, as outputted by nix-build.

- **internal-json**

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- `--no-build-output / -Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- `--max-jobs / -j number`

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- `--cores`

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- `--max-silent-time`

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- `--timeout`

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- `--keep-going / -k`

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- `--keep-failed / -K`

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- `--fallback`

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- `--readonly-mode`

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- `--arg name value`

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`.

When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.  
  system ? builtins.currentSystem  
  ...  
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgnome --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- **--argstr** *name value*

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- **--attr** / **-A** *attrPath*

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression *e*, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- **--expr** / **-E**

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`,

`nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using [paths](#) enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in [impure](#) and [unrestricted](#) evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`

3. /nix/var/nix/profiles/per-user/root/channels

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **NIX_IGNORE_SYMLINK_STORE**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated

as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- **TMPDIR**

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- **NIX_REMOTE**

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- **NIX_SHOW_STATS**

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- **NIX_COUNT_CALLS**

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- **GC_INITIAL_HEAP_SIZE**

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- `XDG_CONFIG_HOME` (default `~/.config`)
- `XDG_STATE_HOME` (default `~/.local/state`)
- `XDG_CACHE_HOME` (default `~/.cache`)

Name

`nix-store --export` - export store paths to a Nix Archive

Synopsis

`nix-store --export paths...`

Description

The operation `--export` writes a serialisation of the specified store paths to standard output in a format that can be imported into another Nix store with `nix-store --import`. This is like `nix-store --dump`, except that the NAR archive produced by that command doesn't contain the necessary meta-information to allow it to be imported into another Nix store (namely, the set of references of the path).

This command does not produce a *closure* of the specified paths, so if a store path references other store paths that are missing in the target Nix store, the import will fail.

Options

The following options are allowed for all `nix-store` operations, but may not always have an effect.

- `--add-root path`

Causes the result of a realisation (`--realise` and `--force-realise`) to be registered as a root of the garbage collector. `path` will be created as a symlink to the resulting store path. In addition, a uniquely named symlink to `path` will be created in `/nix/var/nix/gcroots/auto/`. For instance,

```
$ nix-store --add-root /home/eelco/bla/result --realise ...  
  
$ ls -l /nix/var/nix/gcroots/auto  
lrwxrwxrwx    1 ... 2005-03-13 21:10 dn54lcypm8f8... -> /home/eelco  
/bla/result  
  
$ ls -l /home/eelco/bla/result  
lrwxrwxrwx    1 ... 2005-03-13 21:10 /home/eelco/bla/result -> /nix/store  
/1r11343n6qd4...-f-spot-0.0.10
```

Thus, when `/home/eelco/bla/result` is removed, the GC root in the `auto` directory becomes a dangling symlink and will be ignored by the collector.

Warning

Note that it is not possible to move or rename GC roots, since the symlink in the `auto` directory will still point to the old location.

If there are multiple results, then multiple symlinks will be created by sequentially numbering symlinks beyond the first one (e.g., `foo`, `foo-2`, `foo-3`, and so on).

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose` / `-v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- **0** “Errors only”

Only print messages explaining why the Nix invocation failed.

- **1** “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- **2** “Talkative”

Print more informational messages.

- **3** “Chatty”

Print even more informational messages.

- **4** “Debug”

Print debug information.

- **5** “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to **-v** / **--verbose**.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- **raw**

This is the raw format, as outputted by nix-build.

- **internal-json**

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output / -Q**

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a `default value` (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.  
  system ? builtins.currentSystem  
  ...  
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgnome --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- **--argstr** *name value*

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- **--attr** / **-A** *attrPath*

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression *e*, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- **--expr** / **-E**

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the

built packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under [nix-store --repair-path](#).

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **NIX_IGNORE_SYMLINK_STORE**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [XDG_CONFIG_HOME](#) (default `~/.config`)
- [XDG_STATE_HOME](#) (default `~/.local/state`)

- `XDG_CACHE_HOME` (default `~/cache`)

Examples

To copy a whole closure, do something like:

```
$ nix-store --export $(nix-store --query --requisites paths) > out
```

To import the whole closure again, run:

```
$ nix-store --import < out
```

Name

`nix-store --gc` - run garbage collection

Synopsis

`nix-store --gc [--print-roots | --print-live | --print-dead] [--max-freed bytes]`

Description

Without additional flags, the operation `--gc` performs a garbage collection on the Nix store. That is, all paths in the Nix store not reachable via file system references from a set of “roots”, are deleted.

The following suboperations may be specified:

- `--print-roots`

This operation prints on standard output the set of roots used by the garbage collector.

- `--print-live`

This operation prints on standard output the set of “live” store paths, which are all the store paths reachable from the roots. Live paths should never be deleted, since that would break consistency — it would become possible that applications are installed that reference things that are no longer present in the store.

- `--print-dead`

This operation prints out on standard output the set of “dead” store paths, which is just the opposite of the set of live paths: any path in the store that is not live (with respect to the roots) is dead.

By default, all unreachable paths are deleted. The following options control what gets deleted and in what order:

- `--max-freed bytes`

Keep deleting paths until at least *bytes* bytes have been deleted, then stop. The argument *bytes* can be followed by the multiplicative suffix `K`, `M`, `G` or `T`, denoting KiB, MiB, GiB or TiB units.

The behaviour of the collector is also influenced by the `keep-outputs` and `keep-derivations` settings in the Nix configuration file.

By default, the collector prints the total number of freed bytes when it finishes (or when it is interrupted). With `--print-dead`, it prints the number of bytes that would be freed.

Options

The following options are allowed for all `nix-store` operations, but may not always have an effect.

- `--add-root path`

Causes the result of a realisation (`--realise` and `--force-realise`) to be registered as a root of the garbage collector. *path* will be created as a symlink to the resulting store path. In addition, a uniquely named symlink to *path* will be created in `/nix/var/nix/gcroots/auto`. For instance,

```
$ nix-store --add-root /home/eelco/bla/result --realise ...  
  
$ ls -l /nix/var/nix/gcroots/auto  
lrwxrwxrwx    1 ... 2005-03-13 21:10 dn54lcypm8f8... -> /home/eelco  
/bla/result  
  
$ ls -l /home/eelco/bla/result  
lrwxrwxrwx    1 ... 2005-03-13 21:10 /home/eelco/bla/result -> /nix/store  
/1r11343n6qd4...-f-spot-0.0.10
```

Thus, when `/home/eelco/bla/result` is removed, the GC root in the `auto` directory becomes a dangling symlink and will be ignored by the collector.

Warning

Note that it is not possible to move or rename GC roots, since the symlink in the `auto` directory will still point to the old location.

If there are multiple results, then multiple symlinks will be created by sequentially numbering symlinks beyond the first one (e.g., `foo`, `foo-2`, `foo-3`, and so on).

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose / -v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- `--quiet`

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- `--log-format` *format*

This option can be used to change the output of the log format, with *format* being one of:

- `raw`

This is the raw format, as outputted by nix-build.

- `internal-json`

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- `--no-build-output` / `-Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- `--max-jobs` / `-j` *number*

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If

the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- `--readonly-mode`

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- `--arg name value`

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a `default value` (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.
  system ? builtins.currentSystem
  ...
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgname`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgname --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- `--argstr name value`

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- `--attr / -A attrPath`

Select an attribute from the top-level Nix expression being evaluated. (`nix-env` , `nix-instantiate` , `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression `e`, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- `--expr / -E`

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate` , `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- **NIX_SHELL**

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- **NIX_PATH**

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **NIX_IGNORE_SYMLINK_STORE**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- **TMPDIR**

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- **NIX_REMOTE**

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- **NIX_SHOW_STATS**

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- **NIX_COUNT_CALLS**

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- `XDG_CONFIG_HOME` (default `~/.config`)
- `XDG_STATE_HOME` (default `~/.local/state`)
- `XDG_CACHE_HOME` (default `~/.cache`)

Examples

To delete all unreachable paths, just do:

```
$ nix-store --gc
deleting `/nix/store/kq82idx6g0nyzsp2s14gfsc38npai7lf-cairo-1.0.4.tar.gz.drv'
...
8825586 bytes freed (8.42 MiB)
```

To delete at least 100 MiBs of unreachable paths:

```
$ nix-store --gc --max-freed $((100 * 1024 * 1024))
```

Name

`nix-store --generate-binary-cache-key` - generate key pair to use for a binary cache

Synopsis

`nix-store --generate-binary-cache-key key-name secret-key-file public-key-file`

Description

This command generates an [Ed25519 key pair](#) that can be used to create a signed binary cache. It takes three mandatory parameters:

1. A key name, such as `cache.example.org-1`, that is used to look up keys on the client when it verifies signatures. It can be anything, but it's suggested to use the host name of your cache (e.g. `cache.example.org`) with a suffix denoting the number of the key (to be incremented every time you need to revoke a key).
2. The file name where the secret key is to be stored.
3. The file name where the public key is to be stored.

Options

The following options are allowed for all `nix-store` operations, but may not always have an effect.

- `--add-root path`

Causes the result of a realisation (`--realise` and `--force-realise`) to be registered as a root of the garbage collector. *path* will be created as a symlink to the resulting store path. In addition, a uniquely named symlink to *path* will be created in `/nix/var/nix/gcroots/auto/`. For instance,

```
$ nix-store --add-root /home/eelco/bla/result --realise ...  
  
$ ls -l /nix/var/nix/gcroots/auto  
lrwxrwxrwx    1 ... 2005-03-13 21:10 dn54lcypm8f8... -> /home/eelco  
/bla/result  
  
$ ls -l /home/eelco/bla/result  
lrwxrwxrwx    1 ... 2005-03-13 21:10 /home/eelco/bla/result -> /nix/store  
/1r11343n6qd4...-f-spot-0.0.10
```

Thus, when `/home/eelco/bla/result` is removed, the GC root in the `auto` directory becomes a dangling symlink and will be ignored by the collector.

Warning

Note that it is not possible to move or rename GC roots, since the symlink in the `auto` directory will still point to the old location.

If there are multiple results, then multiple symlinks will be created by sequentially numbering symlinks beyond the first one (e.g., `foo`, `foo-2`, `foo-3`, and so on).

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose` / `-v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- **0** “Errors only”

Only print messages explaining why the Nix invocation failed.

- **1** “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- **2** “Talkative”

Print more informational messages.

- **3** “Chatty”

Print even more informational messages.

- **4** “Debug”

Print debug information.

- **5** “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to **-v** / **--verbose**.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- **raw**

This is the raw format, as outputted by nix-build.

- **internal-json**

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output / -Q**

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a `default value` (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.  
  system ? builtins.currentSystem  
  ...  
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgnome --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- **--argstr** *name value*

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- **--attr** / **-A** *attrPath*

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression *e*, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- **--expr** / **-E**

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the

built packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under [nix-store --repair-path](#).

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **NIX_IGNORE_SYMLINK_STORE**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [XDG_CONFIG_HOME](#) (default `~/.config`)
- [XDG_STATE_HOME](#) (default `~/.local/state`)

- `XDG_CACHE_HOME` (default `~/.cache`)

Name

`nix-store --import` - import Nix Archive into the store

Synopsis

`nix-store --import`

Description

The operation `--import` reads a serialisation of a set of store paths produced by `nix-store --export` from standard input and adds those store paths to the Nix store. Paths that already exist in the Nix store are ignored. If a path refers to another path that doesn't exist in the Nix store, the import fails.

Options

The following options are allowed for all `nix-store` operations, but may not always have an effect.

- `--add-root` *path*

Causes the result of a realisation (`--realise` and `--force-realise`) to be registered as a root of the garbage collector. *path* will be created as a symlink to the resulting store path. In addition, a uniquely named symlink to *path* will be created in `/nix/var/nix/gcroots/auto/`. For instance,

```
$ nix-store --add-root /home/eelco/bla/result --realise ...  
  
$ ls -l /nix/var/nix/gcroots/auto  
lrwxrwxrwx    1 ... 2005-03-13 21:10 dn54lcypm8f8... -> /home/eelco  
/bla/result  
  
$ ls -l /home/eelco/bla/result  
lrwxrwxrwx    1 ... 2005-03-13 21:10 /home/eelco/bla/result -> /nix/store  
/1r11343n6qd4...-f-spot-0.0.10
```

Thus, when `/home/eelco/bla/result` is removed, the GC root in the `auto` directory becomes a dangling symlink and will be ignored by the collector.

Warning

Note that it is not possible to move or rename GC roots, since the symlink in the `auto` directory will still point to the old location.

If there are multiple results, then multiple symlinks will be created by sequentially numbering symlinks beyond the first one (e.g., `foo`, `foo-2`, `foo-3`, and so on).

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose` / `-v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- **0** “Errors only”

Only print messages explaining why the Nix invocation failed.

- **1** “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- **2** “Talkative”

Print more informational messages.

- **3** “Chatty”

Print even more informational messages.

- **4** “Debug”

Print debug information.

- **5** “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to **-v** / **--verbose**.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- **raw**

This is the raw format, as outputted by nix-build.

- **internal-json**

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output / -Q**

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a `default value` (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.  
  system ? builtins.currentSystem  
  ...  
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgnome --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- **--argstr** *name value*

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- **--attr** / **-A** *attrPath*

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression *e*, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- **--expr** / **-E**

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the

built packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under [nix-store --repair-path](#).

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **NIX_IGNORE_SYMLINK_STORE**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [XDG_CONFIG_HOME](#) (default `~/.config`)
- [XDG_STATE_HOME](#) (default `~/.local/state`)

- `XDG_CACHE_HOME` (default `~/.cache`)

Name

`nix-store --load-db` - import Nix database

Synopsis

`nix-store --load-db`

Description

The operation `--load-db` reads a dump of the Nix database created by `--dump-db` from standard input and loads it into the Nix database.

Options

The following options are allowed for all `nix-store` operations, but may not always have an effect.

- `--add-root path`

Causes the result of a realisation (`--realise` and `--force-realise`) to be registered as a root of the garbage collector. *path* will be created as a symlink to the resulting store path. In addition, a uniquely named symlink to *path* will be created in `/nix/var/nix/gcroots/auto/`. For instance,

```
$ nix-store --add-root /home/eelco/bla/result --realise ...
```

```
$ ls -l /nix/var/nix/gcroots/auto
lrwxrwxrwx    1 ... 2005-03-13 21:10 dn54lcypm8f8... -> /home/eelco
/bla/result
```

```
$ ls -l /home/eelco/bla/result
lrwxrwxrwx    1 ... 2005-03-13 21:10 /home/eelco/bla/result -> /nix/store
/1r11343n6qd4...-f-spot-0.0.10
```

Thus, when `/home/eelco/bla/result` is removed, the GC root in the `auto` directory becomes a dangling symlink and will be ignored by the collector.

Warning

Note that it is not possible to move or rename GC roots, since the symlink in the `auto` directory will still point to the old location.

If there are multiple results, then multiple symlinks will be created by sequentially numbering symlinks beyond the first one (e.g., `foo`, `foo-2`, `foo-3`, and so on).

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose / -v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- `0` “Errors only”

Only print messages explaining why the Nix invocation failed.

- `1` “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- `2` “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- `raw`

This is the raw format, as outputted by nix-build.

- `internal-json`

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output / -Q**

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in `Nixpkgs`, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in `Nixpkgs` is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.
  system ? builtins.currentSystem
  ...
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgname`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgname --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- `--argstr name value`

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- `--attr / -A attrPath`

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression *e*, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- `--expr / -E`

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- [--repair](#)

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under [nix-store --repair-path](#).

Common Environment Variables

Most Nix commands interpret the following environment variables:

- [IN_NIX_SHELL](#)

Indicator that tells if the current environment was set up by [nix-shell](#). It can have the values [pure](#) or [impure](#).

- [NIX_PATH](#)

A colon-separated list of directories used to look up the location of Nix expressions using [paths](#) enclosed in angle brackets (i.e., [`<path>`](#)), e.g. [`/home/eelco/Dev:/etc/nixos`](#). It can be extended using the [-I option](#).

If [NIX_PATH](#) is not set at all, Nix will fall back to the following list in [impure](#) and [unrestricted](#) evaluation mode:

1. [`\$HOME/.nix-defexpr/channels`](#)
2. [`nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`](#)
3. [`/nix/var/nix/profiles/per-user/root/channels`](#)

If [NIX_PATH](#) is set to an empty string, resolving search paths will always fail. For example, attempting to use [`<nixpkgs>`](#) will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- [NIX_IGNORE_SYMLINK_STORE](#)

Normally, the Nix store directory (typically [`/nix/store`](#)) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with [`/nix/store`](#) resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where [`/nix/store`](#) resolves differently. If you are sure that you’re not going to do that, you can set [NIX_IGNORE_SYMLINK_STORE](#) to [1](#).

Note that if you're symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you're better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- **TMPDIR**

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- **NIX_REMOTE**

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [`NIX_SHOW_STATS`](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [`NIX_COUNT_CALLS`](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [`GC_INITIAL_HEAP_SIZE`](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [`XDG_CONFIG_HOME`](#) (default `~/.config`)
- [`XDG_STATE_HOME`](#) (default `~/.local/state`)
- [`XDG_CACHE_HOME`](#) (default `~/.cache`)

Name

`nix-store --optimise` - reduce disk space usage

Synopsis

`nix-store --optimise`

Description

The operation `--optimise` reduces Nix store disk space usage by finding identical files in the store and hard-linking them to each other. It typically reduces the size of the store by something like 25-35%. Only regular files and symlinks are hard-linked in this manner. Files are considered identical when they have the same NAR archive serialisation: that is, regular files must have the same contents and permission (executable or non-executable), and symlinks must have the same contents.

After completion, or when the command is interrupted, a report on the achieved savings is printed on standard error.

Use `-vv` or `-vvv` to get some progress indication.

Options

The following options are allowed for all `nix-store` operations, but may not always have an effect.

- `--add-root path`

Causes the result of a realisation (`--realise` and `--force-realise`) to be registered as a root of the garbage collector. `path` will be created as a symlink to the resulting store path. In addition, a uniquely named symlink to `path` will be created in `/nix/var/nix/gcroots/auto/`. For instance,

```
$ nix-store --add-root /home/eelco/bla/result --realise ...  
  
$ ls -l /nix/var/nix/gcroots/auto  
lrwxrwxrwx    1 ... 2005-03-13 21:10 dn54lcypm8f8... -> /home/eelco  
/bla/result  
  
$ ls -l /home/eelco/bla/result  
lrwxrwxrwx    1 ... 2005-03-13 21:10 /home/eelco/bla/result -> /nix/store  
/1r11343n6qd4...-f-spot-0.0.10
```

Thus, when `/home/eelco/bla/result` is removed, the GC root in the `auto` directory becomes a dangling symlink and will be ignored by the collector.

Warning

Note that it is not possible to move or rename GC roots, since the symlink in the `auto` directory will still point to the old location.

If there are multiple results, then multiple symlinks will be created by sequentially numbering symlinks beyond the first one (e.g., `foo`, `foo-2`, `foo-3`, and so on).

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose` / `-v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- **0** “Errors only”

Only print messages explaining why the Nix invocation failed.

- **1** “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- **2** “Talkative”

Print more informational messages.

- **3** “Chatty”

Print even more informational messages.

- **4** “Debug”

Print debug information.

- **5** “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to **-v** / **--verbose**.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- **raw**

This is the raw format, as outputted by nix-build.

- **internal-json**

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output / -Q**

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a `default value` (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.  
  system ? builtins.currentSystem  
  ...  
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgnome --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- **--argstr** *name value*

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- **--attr** / **-A** *attrPath*

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression *e*, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- **--expr** / **-E**

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the

built packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under [nix-store --repair-path](#).

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **NIX_IGNORE_SYMLINK_STORE**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [XDG_CONFIG_HOME](#) (default `~/.config`)
- [XDG_STATE_HOME](#) (default `~/.local/state`)

- `XDG_CACHE_HOME` (default `~/.cache`)

Example

```
$ nix-store --optimise
hashing files in `/nix/store/qhqx7l2f1kmwihc9bnxs7rc159hsxnf3-gcc-4.1.1'
...
541838819 bytes (516.74 MiB) freed by hard-linking 54143 files;
there are 114486 files with equal contents out of 215894 files in total
```

Name

`nix-store --print-env` - print the build environment of a derivation

Synopsis

`nix-store --print-env drvpath`

Description

The operation `--print-env` prints out the environment of a derivation in a format that can be evaluated by a shell. The command line arguments of the builder are placed in the variable `_args`.

Options

The following options are allowed for all `nix-store` operations, but may not always have an effect.

- `--add-root path`

Causes the result of a realisation (`--realise` and `--force-realise`) to be registered as a root of the garbage collector. *path* will be created as a symlink to the resulting store path. In addition, a uniquely named symlink to *path* will be created in `/nix/var/nix/gcroots/auto/`. For instance,

```
$ nix-store --add-root /home/eelco/bla/result --realise ...  
  
$ ls -l /nix/var/nix/gcroots/auto  
lrwxrwxrwx    1 ... 2005-03-13 21:10 dn54lcypm8f8... -> /home/eelco  
/bla/result  
  
$ ls -l /home/eelco/bla/result  
lrwxrwxrwx    1 ... 2005-03-13 21:10 /home/eelco/bla/result -> /nix/store  
/1r11343n6qd4...-f-spot-0.0.10
```

Thus, when `/home/eelco/bla/result` is removed, the GC root in the `auto` directory becomes a dangling symlink and will be ignored by the collector.

Warning

Note that it is not possible to move or rename GC roots, since the symlink in the `auto` directory will still point to the old location.

If there are multiple results, then multiple symlinks will be created by sequentially numbering symlinks beyond the first one (e.g., `foo`, `foo-2`, `foo-3`, and so on).

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose` / `-v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- `raw`

This is the raw format, as outputted by nix-build.

- `internal-json`

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output** / `-Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg** *name value*

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.
  system ? builtins.currentSystem
  ...
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value

`builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgname --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- `--argstr name value`

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- `--attr / -A attrPath`

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path* `attrPath` is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression `e`, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- `--expr / -E`

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option `name` to `value`. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- `NIX_IGNORE_SYMLINK_STORE`

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system

than the root file system, on Linux you're better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- **TMPDIR**

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- **NIX_REMOTE**

This variable should be set to `daemon` if you want to use the Nix daemon to execute

Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [`NIX_SHOW_STATS`](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [`NIX_COUNT_CALLS`](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [`GC_INITIAL_HEAP_SIZE`](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [`XDG_CONFIG_HOME`](#) (default `~/.config`)
- [`XDG_STATE_HOME`](#) (default `~/.local/state`)
- [`XDG_CACHE_HOME`](#) (default `~/.cache`)

Example

```
$ nix-store --print-env $(nix-instantiate '<nixpkgs>' -A firefox)
...
export src; src='/nix/store/plpj7qrwcz94z2psh6fchsi7s8yihc7k-firefox-
12.0.source.tar.bz2'
export stdenv; stdenv='/nix/store/7c8asx3yfrg5dg1gzhzyq2236zfgibnn-stdenv'
export system; system='x86_64-linux'
export _args; _args='-e /nix/store/9kr1zvny65gdc8s7kpb6lkx8cd02c25c-default-
builder.sh'
```

Name

`nix-store --query` - display information about store paths

Synopsis

```
nix-store { --query | -q }{ --outputs | --requisites | -R | --references |
--referrers | --referrers-closure | --deriver | -d | --valid-deriviers | --graph |
--tree | --binding name | -b name | --hash | --size | --roots }[ --use-output ]
[-u][--force-realise][-f] paths...
```

Description

The operation `--query` displays various bits of information about the store paths . The queries are described below. At most one query can be specified. The default query is `--outputs` .

The paths *paths* may also be symlinks from outside of the Nix store, to the Nix store. In that case, the query is applied to the target of the symlink.

Common query options

- `--use-output`; `-u`

For each argument to the query that is a [store derivation](#), apply the query to the output path of the derivation instead.

- `--force-realise`; `-f`

Realise each argument to the query first (see [`nix-store --realise`](#)).

Queries

- `--outputs`

Prints out the [output paths](#) of the store derivations *paths*. These are the paths that will be produced when the derivation is built.

- `--requisites ; -R`

Prints out the [closure](#) of the store path *paths*.

This query has one option:

- `--include-outputs` Also include the existing output paths of [store derivations](#), and their closures.

This query can be used to implement various kinds of deployment. A *source deployment* is obtained by distributing the closure of a store derivation. A *binary deployment* is obtained by distributing the closure of an output path. A *cache deployment* (combined source/binary deployment, including binaries of build-time-only dependencies) is obtained by distributing the closure of a store derivation and specifying the option `--include-outputs`.

- `--references`

Prints the set of [references](#) of the store paths *paths*, that is, their immediate dependencies. (For *all* dependencies, use `--requisites`.)

- `--referrers`

Prints the set of *referrers* of the store paths *paths*, that is, the store paths currently existing in the Nix store that refer to one of *paths*. Note that contrary to the references, the set of referrers is not constant; it can change as store paths are added or removed.

- `--referrers-closure`

Prints the closure of the set of store paths *paths* under the referrers relation; that is, all store paths that directly or indirectly refer to one of *paths*. These are all the path currently in the Nix store that are dependent on *paths*.

- `--deriver ; -d`

Prints the [deriver](#) that was used to build the store paths *paths*. If the path has no deriver (e.g., if it is a source file), or if the deriver is not known (e.g., in the case of a binary-only deployment), the string `unknown-deriver` is printed. The returned deriver is not guaranteed to exist in the local store, for example when *paths* were substituted from a binary cache. Use `--valid-derivars` instead to obtain valid paths only.

- `--valid-derivars`

Prints a set of derivation files (`.drv`) which are supposed produce said paths when realized. Might print nothing, for example for source paths or paths subsituted from a binary cache.

- `--graph`

Prints the references graph of the store paths *paths* in the format of the [dot](#) tool of

AT&T's [Graphviz package](#). This can be used to visualise dependency graphs. To obtain a build-time dependency graph, apply this to a store derivation. To obtain a runtime dependency graph, apply it to an output path.

- **--tree**

Prints the references graph of the store paths *paths* as a nested ASCII tree. References are ordered by descending closure size; this tends to flatten the tree, making it more readable. The query only recurses into a store path when it is first encountered; this prevents a blowup of the tree representation of the graph.

- **--graphml**

Prints the references graph of the store paths *paths* in the [GraphML](#) file format. This can be used to visualise dependency graphs. To obtain a build-time dependency graph, apply this to a [store derivation](#). To obtain a runtime dependency graph, apply it to an output path.

- **--binding** *name*; **-b** *name*

Prints the value of the attribute *name* (i.e., environment variable) of the [store derivations](#) *paths*. It is an error for a derivation to not have the specified attribute.

- **--hash**

Prints the SHA-256 hash of the contents of the store paths *paths* (that is, the hash of the output of `nix-store --dump` on the given paths). Since the hash is stored in the Nix database, this is a fast operation.

- **--size**

Prints the size in bytes of the contents of the store paths *paths* — to be precise, the size of the output of `nix-store --dump` on the given paths. Note that the actual disk space required by the store paths may be higher, especially on filesystems with large cluster sizes.

- **--roots**

Prints the garbage collector roots that point, directly or indirectly, at the store paths *paths*.

Options

The following options are allowed for all `nix-store` operations, but may not always have an effect.

- **--add-root** *path*

Causes the result of a realisation (`--realise` and `--force-realise`) to be registered as a root of the garbage collector. *path* will be created as a symlink to the resulting store path. In addition, a uniquely named symlink to *path* will be created in `/nix/var/nix/gcroots/auto`. For instance,

```
$ nix-store --add-root /home/eelco/bla/result --realise ...  
  
$ ls -l /nix/var/nix/gcroots/auto  
lrwxrwxrwx 1 ... 2005-03-13 21:10 dn54lcypm8f8... -> /home/eelco  
/bla/result  
  
$ ls -l /home/eelco/bla/result  
lrwxrwxrwx 1 ... 2005-03-13 21:10 /home/eelco/bla/result -> /nix/store  
/1r11343n6qd4...-f-spot-0.0.10
```

Thus, when `/home/eelco/bla/result` is removed, the GC root in the `auto` directory becomes a dangling symlink and will be ignored by the collector.

Warning

Note that it is not possible to move or rename GC roots, since the symlink in the `auto` directory will still point to the old location.

If there are multiple results, then multiple symlinks will be created by sequentially numbering symlinks beyond the first one (e.g., `foo`, `foo-2`, `foo-3`, and so on).

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose` / `-v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- **0** “Errors only”

Only print messages explaining why the Nix invocation failed.

- **1** “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- **2** “Talkative”

Print more informational messages.

- **3** “Chatty”

Print even more informational messages.

- **4** “Debug”

Print debug information.

- **5** “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to **-v** / **--verbose**.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- **raw**

This is the raw format, as outputted by nix-build.

- **internal-json**

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- `--no-build-output / -Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- `--max-jobs / -j number`

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- `--cores`

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- `--max-silent-time`

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- `--timeout`

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- `--keep-going / -k`

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- `--keep-failed / -K`

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- `--fallback`

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- `--readonly-mode`

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- `--arg name value`

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`.

When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.  
  system ? builtins.currentSystem  
  ...  
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgnome --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- **--argstr** *name value*

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- **--attr** / **-A** *attrPath*

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression *e*, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- **--expr** / **-E**

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`,

`nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`

3. /nix/var/nix/profiles/per-user/root/channels

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **NIX_IGNORE_SYMLINK_STORE**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated

as if it was read from a Nix configuration file. Settings are separated by the newline character.

- [NIX_USER_CONF_FILES](#)

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- `XDG_CONFIG_HOME` (default `~/.config`)
- `XDG_STATE_HOME` (default `~/.local/state`)
- `XDG_CACHE_HOME` (default `~/.cache`)

Examples

Print the closure (runtime dependencies) of the `svn` program in the current user environment:

```
$ nix-store --query --requisites $(which svn)
/nix/store/5mbglq5ldqld8sj57273aljwkfvj22mc-subversion-1.1.4
/nix/store/9lz9yc6zgmc0vlqmn2ipcpkjlmbi51vv-glibc-2.3.4
...
...
```

Print the build-time dependencies of `svn`:

```
$ nix-store --query --requisites $(nix-store --query --deriver $(which svn))
/nix/store/02iizgn86m42q905rddvg4ja975bk2i4-grep-2.5.1.tar.bz2.drv
/nix/store/07a2bzxmzwz5hp58nf03pahrv2ygwgs3-gcc-wrapper.sh
/nix/store/0ma7c9wsbaxahwwl04gbw3fc806ski4-glibc-2.3.4.drv
... lots of other paths ...
```

The difference with the previous example is that we ask the closure of the derivation (`-qd`), not the closure of the output path that contains `svn`.

Show the build-time dependencies as a tree:

```
$ nix-store --query --tree $(nix-store --query --deriver $(which svn))
/nix/store/7i5082kfb6yjbqdbiwdhhza0am2xvh6c-subversion-1.1.4.drv
+-- /nix/store/d8afh10z72n8l1cr5w42366abiblgn54-builder.sh
+-- /nix/store/fmzxmpjx2lh849ph0l36snfj9zdibw67-bash-3.0.drv
|   +-- /nix/store/570hmhmrx3v57605cqg9yfvvyh0nnb8k8-bash
|   +-- /nix/store/p3srsbd8dx44v2pg6nbnszab5mcwx03v-builder.sh
...
...
```

Show all paths that depend on the same OpenSSL library as `svn`:

```
$ nix-store --query --referrers $(nix-store --query --binding openssl $(nix-store --query --deriver $(which svn)))
/nix/store/23ny9l9wixx21632y2wi4p585qhva1q8-sylpheed-1.0.0
/nix/store/5mbglq5ldqld8sj57273aljwkfvj22mc-subversion-1.1.4
/nix/store/dpmvp969yhdqs7lm2r1a3gng7pyq6vy4-subversion-1.1.3
/nix/store/l51240xqsgg8a7yrbqdx1rfzyv6l26fx-lynx-2.8.5
```

Show all paths that directly or indirectly depend on the Glibc (C library) used by `svn`:

```
$ nix-store --query --referrers-closure $(ldd $(which svn) | grep /libc.so | awk '{print $3}')
/nix/store/034a6h4vpz9kds5r6kzb9lhh81mscw43-libgnomeprintui-2.8.2
/nix/store/15l3yi0d45prm7a82pcrknxdh6nzmxza-gawk-3.1.4
...
```

Note that `ldd` is a command that prints out the dynamic libraries used by an ELF executable.

Make a picture of the runtime dependency graph of the current user environment:

```
$ nix-store --query --graph ~/.nix-profile | dot -Tps > graph.ps
$ gv graph.ps
```

Show every garbage collector root that points to a store path that depends on `svn`:

```
$ nix-store --query --roots $(which svn)
/nix/var/nix/profiles/default-81-link
/nix/var/nix/profiles/default-82-link
/home/eelco/.local/state/nix/profiles/profile-97-link
```

Name

`nix-store --read-log` - print build log

Synopsis

`nix-store { --read-log | -l } paths...`

Description

The operation `--read-log` prints the build log of the specified store paths on standard output. The build log is whatever the builder of a derivation wrote to standard output and standard error. If a store path is not a derivation, the deriver of the store path is used.

Build logs are kept in `/nix/var/log/nix/drvs`. However, there is no guarantee that a build log is available for any particular store path. For instance, if the path was downloaded as a pre-built binary through a substitute, then the log is unavailable.

Options

The following options are allowed for all `nix-store` operations, but may not always have an effect.

- `--add-root path`

Causes the result of a realisation (`--realise` and `--force-realise`) to be registered as a root of the garbage collector. `path` will be created as a symlink to the resulting store path. In addition, a uniquely named symlink to `path` will be created in `/nix/var/nix/gcroots/auto/`. For instance,

```
$ nix-store --add-root /home/eelco/bla/result --realise ...  
  
$ ls -l /nix/var/nix/gcroots/auto  
lrwxrwxrwx    1 ... 2005-03-13 21:10 dn54lcypm8f8... -> /home/eelco  
/bla/result  
  
$ ls -l /home/eelco/bla/result  
lrwxrwxrwx    1 ... 2005-03-13 21:10 /home/eelco/bla/result -> /nix/store  
/1r11343n6qd4...-f-spot-0.0.10
```

Thus, when `/home/eelco/bla/result` is removed, the GC root in the `auto` directory becomes a dangling symlink and will be ignored by the collector.

Warning

Note that it is not possible to move or rename GC roots, since the symlink in the `auto` directory will still point to the old location.

If there are multiple results, then multiple symlinks will be created by sequentially numbering symlinks beyond the first one (e.g., `foo`, `foo-2`, `foo-3`, and so on).

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose` / `-v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- `raw`

This is the raw format, as outputted by nix-build.

- `internal-json`

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output / -Q**

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a `default value` (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.  
  system ? builtins.currentSystem  
  ...  
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgnome --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- **--argstr** *name value*

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- **--attr** / **-A** *attrPath*

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression *e*, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- **--expr** / **-E**

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the

built packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under [nix-store --repair-path](#).

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **NIX_IGNORE_SYMLINK_STORE**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [XDG_CONFIG_HOME](#) (default `~/.config`)
- [XDG_STATE_HOME](#) (default `~/.local/state`)

- `XDG_CACHE_HOME` (default `~/.cache`)

Example

```
$ nix-store --read-log $(which ktorrent)
building /nix/store/dhc73pvzpnzxhdgpimsd9sw39di66ph1-ktorrent-2.2.1
unpacking sources
unpacking source archive /nix/store/p8n1jpqs27mgkjh07pb5269717nzf5f8-ktorrent-
2.2.1.tar.gz
ktorrent-2.2.1/
ktorrent-2.2.1/NEWS
...
...
```

Name

`nix-store --realise` - build or fetch store objects

Synopsis

`nix-store { --realise | -r } paths... [--dry-run]`

Description

Each of *paths* is processed as follows:

- If the path leads to a [store derivation](#):
 1. If it is not [valid](#), substitute the store derivation file itself.
 2. Realise its [output paths](#):
 - Try to fetch from [substituters](#) the [store objects](#) associated with the output paths in the store derivation's [closure](#).
 - With [content-addressed derivations](#) (experimental): Determine the output paths to realise by querying content-addressed realisation entries in the [Nix database](#).
 - For any store paths that cannot be substituted, produce the required store objects. This involves first realising all outputs of the derivation's dependencies and then running the derivation's [builder](#) executable.
 - Otherwise, and if the path is not already valid: Try to fetch the associated [store objects](#) in the path's [closure](#) from [substituters](#).

If no substitutes are available and no store derivation is given, realisation fails.

The resulting paths are printed on standard output. For non-derivation arguments, the argument itself is printed.

Special exit codes for build failure

1xx status codes are used when requested builds failed. The following codes are in use:

- [100](#) Generic build failure

The builder process returned with a non-zero exit code.

- [101 Build timeout](#)

The build was aborted because it did not complete within the specified `timeout`.

- [102 Hash mismatch](#)

The build output was rejected because it does not match the `outputHash` attribute of the derivation.

- [104 Not deterministic](#)

The build succeeded in check mode but the resulting output is not binary reproducible.

With the `--keep-going` flag it's possible for multiple failures to occur. In this case the 1xx status codes are or combined using [bitwise OR](#).

```
0b1100100
     ^^^^
    |||`- timeout
    ||`-- output hash mismatch
    |`--- build failure
    `---- not deterministic
```

Options

- [--dry-run](#)

Print on standard error a description of what packages would be built or downloaded, without actually performing the operation.

- [--ignore-unknown](#)

If a non-derivation path does not have a substitute, then silently ignore it.

- [--check](#)

This option allows you to check whether a derivation is deterministic. It rebuilds the specified derivation and checks whether the result is bitwise-identical with the existing outputs, printing an error if that's not the case. The outputs of the specified derivation must already exist. When used with [-K](#), if an output path is not identical to the corresponding output from the previous build, the new output path is left in `/nix/store/name.check`.

Options

The following options are allowed for all `nix-store` operations, but may not always have an effect.

- `--add-root path`

Causes the result of a realisation (`--realise` and `--force-realise`) to be registered as a root of the garbage collector. *path* will be created as a symlink to the resulting store path. In addition, a uniquely named symlink to *path* will be created in `/nix/var/nix/gcroots/auto/`. For instance,

```
$ nix-store --add-root /home/eelco/bla/result --realise ...  
  
$ ls -l /nix/var/nix/gcroots/auto  
lrwxrwxrwx    1 ... 2005-03-13 21:10 dn54lcypm8f8... -> /home/eelco  
/bla/result  
  
$ ls -l /home/eelco/bla/result  
lrwxrwxrwx    1 ... 2005-03-13 21:10 /home/eelco/bla/result -> /nix/store  
/1r11343n6qd4...-f-spot-0.0.10
```

Thus, when `/home/eelco/bla/result` is removed, the GC root in the `auto` directory becomes a dangling symlink and will be ignored by the collector.

Warning

Note that it is not possible to move or rename GC roots, since the symlink in the `auto` directory will still point to the old location.

If there are multiple results, then multiple symlinks will be created by sequentially numbering symlinks beyond the first one (e.g., `foo`, `foo-2`, `foo-3`, and so on).

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose` / `-v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- `--quiet`

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- `--log-format` *format*

This option can be used to change the output of the log format, with *format* being one of:

- **raw**

This is the raw format, as outputted by nix-build.

- **internal-json**

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the **msg**-field) can change between releases.

- **bar**

Only display a progress bar during the builds.

- **bar-with-logs**

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output / -Q**

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in **prefix/nix/var/log/nix**.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify **auto** to use the number of CPUs in the system. The default is specified by the **max-jobs** configuration setting, which itself defaults to **1**. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to **0** disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the **NIX_BUILD_CORES** environment variable in the invocation of

builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg** *name value*

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.
  system ? builtins.currentSystem
  ...
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgnome --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- **--argstr** *name value*

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- **--attr / -A** *attrPath*

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression *e*, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- `--expr / -E`

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions

using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **`NIX_IGNORE_SYMLINK_STORE`**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **`NIX_STORE_DIR`**

Overrides the location of the Nix store (default `prefix/store`).

- **`NIX_DATA_DIR`**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **`NIX_LOG_DIR`**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **`NIX_STATE_DIR`**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- **TMPDIR**

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- **NIX_REMOTE**

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- **NIX_SHOW_STATS**

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- **NIX_COUNT_CALLS**

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- **GC_INITIAL_HEAP_SIZE**

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- `XDG_CONFIG_HOME` (default `~/.config`)
- `XDG_STATE_HOME` (default `~/.local/state`)
- `XDG_CACHE_HOME` (default `~/.cache`)

Examples

This operation is typically used to build [store derivations](#) produced by `nix-instantiate`:

```
$ nix-store --realise $(nix-instantiate ./test.nix)
/nix/store/31axcgrlbfsxzmf1gyj1bf62hvkby2-aterm-2.3.1
```

This is essentially what `nix-build` does.

To test whether a previously-built derivation is deterministic:

```
$ nix-build '<nixpkgs>' --attr hello --check -K
```

Use `nix-store --read-log` to show the stderr and stdout of a build:

```
$ nix-store --read-log $(nix-instantiate ./test.nix)
```

Name

`nix --repair-path` - re-download path from substituter

Synopsis

`nix-store --repair-path paths...`

Description

The operation `--repair-path` attempts to “repair” the specified paths by redownloading them using the available substituters. If no substitutes are available, then repair is not possible.

Warning

During repair, there is a very small time window during which the old path (if it exists) is moved out of the way and replaced with the new path. If repair is interrupted in between, then the system may be left in a broken state (e.g., if the path contains a critical system component like the GNU C Library).

Example

```
$ nix-store --verify-path /nix/store/dj7a81wsm1ijwwpkks3725661h3263p5-glibc-2.13
path `/nix/store/dj7a81wsm1ijwwpkks3725661h3263p5-glibc-2.13' was modified!
  expected hash
`2db57715ae90b7e31ff1f2ecb8c12ec1cc43da920efcbe3b22763f36a1861588',
  got `481c5aa5483ebc97c20457bb8bca24deea56550d3985cda0027f67fe54b808e4'

$ nix-store --repair-path /nix/store/dj7a81wsm1ijwwpkks3725661h3263p5-glibc-2.13
fetching path `/nix/store/d7a81wsm1ijwwpkks3725661h3263p5-glibc-2.13'...
...
```

Name

`nix-store --restore` - extract a Nix archive

Synopsis

`nix-store --restore` *path*

Description

The operation `--restore` unpacks a NAR archive to *path*, which must not already exist. The archive is read from standard input.

Options

The following options are allowed for all `nix-store` operations, but may not always have an effect.

- `--add-root` *path*

Causes the result of a realisation (`--realise` and `--force-realise`) to be registered as a root of the garbage collector. *path* will be created as a symlink to the resulting store path. In addition, a uniquely named symlink to *path* will be created in `/nix/var/nix/gcroots/auto/`. For instance,

```
$ nix-store --add-root /home/eelco/bla/result --realise ...  
  
$ ls -l /nix/var/nix/gcroots/auto  
lrwxrwxrwx    1 ... 2005-03-13 21:10 dn54lcypm8f8... -> /home/eelco  
/bla/result  
  
$ ls -l /home/eelco/bla/result  
lrwxrwxrwx    1 ... 2005-03-13 21:10 /home/eelco/bla/result -> /nix/store  
/1r11343n6qd4...-f-spot-0.0.10
```

Thus, when `/home/eelco/bla/result` is removed, the GC root in the `auto` directory becomes a dangling symlink and will be ignored by the collector.

Warning

Note that it is not possible to move or rename GC roots, since the symlink in the `auto` directory will still point to the old location.

If there are multiple results, then multiple symlinks will be created by sequentially numbering symlinks beyond the first one (e.g., `foo`, `foo-2`, `foo-3`, and so on).

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose` / `-v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- `raw`

This is the raw format, as outputted by nix-build.

- `internal-json`

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg` -field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output** / `-Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that

the builder's standard output and error are always written to a log file in `prefix/nix` `/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is

printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.
  system ? builtins.currentSystem
  ...
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgname`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgname --arg system \"i686-freebsd\"`.

(Note that since the argument is a Nix string literal, you have to escape the quotes.)

- `--argstr name value`

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- `--attr / -A attrPath`

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path* `attrPath` is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression `e`, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- `--expr / -E`

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that

this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- `NIX_IGNORE_SYMLINK_STORE`

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix  
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- **TMPDIR**

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- **NIX_REMOTE**

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path`

`/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [XDG_CONFIG_HOME](#) (default `~/.config`)
- [XDG_STATE_HOME](#) (default `~/.local/state`)
- [XDG_CACHE_HOME](#) (default `~/.cache`)

Name

`nix-store --serve` - serve local Nix store over SSH

Synopsis

`nix-store --serve [--write]`

Description

The operation `--serve` provides access to the Nix store over stdin and stdout, and is intended to be used as a means of providing Nix store access to a restricted ssh user.

The following flags are available:

- `--write`

Allow the connected client to request the realization of derivations. In effect, this can be used to make the host act as a remote builder.

Options

The following options are allowed for all `nix-store` operations, but may not always have an effect.

- `--add-root path`

Causes the result of a realisation (`--realise` and `--force-realise`) to be registered as a root of the garbage collector. *path* will be created as a symlink to the resulting store path. In addition, a uniquely named symlink to *path* will be created in `/nix/var/nix/gcroots/auto/`. For instance,

```
$ nix-store --add-root /home/eelco/bla/result --realise ...  
  
$ ls -l /nix/var/nix/gcroots/auto  
lrwxrwxrwx 1 ... 2005-03-13 21:10 dn54lcypm8f8... -> /home/eelco  
/bla/result  
  
$ ls -l /home/eelco/bla/result  
lrwxrwxrwx 1 ... 2005-03-13 21:10 /home/eelco/bla/result -> /nix/store  
/1r11343n6qd4...-f-spot-0.0.10
```

Thus, when `/home/eelco/bla/result` is removed, the GC root in the `auto` directory becomes a dangling symlink and will be ignored by the collector.

Warning

Note that it is not possible to move or rename GC roots, since the symlink in the `auto` directory will still point to the old location.

If there are multiple results, then multiple symlinks will be created by sequentially numbering symlinks beyond the first one (e.g., `foo`, `foo-2`, `foo-3`, and so on).

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose` / `-v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- **0** “Errors only”

Only print messages explaining why the Nix invocation failed.

- **1** “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- **2** “Talkative”

Print more informational messages.

- **3** “Chatty”

Print even more informational messages.

- **4** “Debug”

Print debug information.

- **5** “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to **-v** / **--verbose**.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- **raw**

This is the raw format, as outputted by nix-build.

- **internal-json**

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output / -Q**

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a `default value` (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.  
  system ? builtins.currentSystem  
  ...  
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgnome --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- **--argstr** *name value*

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- **--attr** / **-A** *attrPath*

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression *e*, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- **--expr** / **-E**

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the

built packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under [nix-store --repair-path](#).

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **NIX_IGNORE_SYMLINK_STORE**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [XDG_CONFIG_HOME](#) (default `~/.config`)
- [XDG_STATE_HOME](#) (default `~/.local/state`)

- `XDG_CACHE_HOME` (default `~/cache`)

Examples

To turn a host into a build server, the `authorized_keys` file can be used to provide build access to a given SSH public key:

```
$ cat <<EOF >>/root/.ssh/authorized_keys
command="nice -n20 nix-store --serve --write" ssh-rsa AAAAB3NzaC1yc2EAAA...EOF
```

Name

`nix-store --verify-path` - check path contents against Nix database

Synopsis

`nix-store --verify-path paths...`

Description

The operation `--verify-path` compares the contents of the given store paths to their cryptographic hashes stored in Nix's database. For every changed path, it prints a warning message. The exit status is 0 if no path has changed, and 1 otherwise.

Options

The following options are allowed for all `nix-store` operations, but may not always have an effect.

- `--add-root path`

Causes the result of a realisation (`--realise` and `--force-realise`) to be registered as a root of the garbage collector. *path* will be created as a symlink to the resulting store path. In addition, a uniquely named symlink to *path* will be created in `/nix/var/nix/gcroots/auto/`. For instance,

```
$ nix-store --add-root /home/eelco/bla/result --realise ...  
  
$ ls -l /nix/var/nix/gcroots/auto  
lrwxrwxrwx    1 ... 2005-03-13 21:10 dn54lcypm8f8... -> /home/eelco  
/bla/result  
  
$ ls -l /home/eelco/bla/result  
lrwxrwxrwx    1 ... 2005-03-13 21:10 /home/eelco/bla/result -> /nix/store  
/1r11343n6qd4...-f-spot-0.0.10
```

Thus, when `/home/eelco/bla/result` is removed, the GC root in the `auto` directory becomes a dangling symlink and will be ignored by the collector.

Warning

Note that it is not possible to move or rename GC roots, since the symlink in the `auto` directory will still point to the old location.

If there are multiple results, then multiple symlinks will be created by sequentially numbering symlinks beyond the first one (e.g., `foo`, `foo-2`, `foo-3`, and so on).

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose` / `-v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- `raw`

This is the raw format, as outputted by nix-build.

- `internal-json`

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg` -field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output** / `-Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg** *name value*

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.
  system ? builtins.currentSystem
  ...
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value

`builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgname --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- `--argstr name value`

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- `--attr / -A attrPath`

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path* `attrPath` is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression `e`, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- `--expr / -E`

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- `NIX_IGNORE_SYMLINK_STORE`

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system

than the root file system, on Linux you're better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- **TMPDIR**

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- **NIX_REMOTE**

This variable should be set to `daemon` if you want to use the Nix daemon to execute

Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [`NIX_SHOW_STATS`](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [`NIX_COUNT_CALLS`](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [`GC_INITIAL_HEAP_SIZE`](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [`XDG_CONFIG_HOME`](#) (default `~/.config`)
- [`XDG_STATE_HOME`](#) (default `~/.local/state`)
- [`XDG_CACHE_HOME`](#) (default `~/.cache`)

Example

To verify the integrity of the `svn` command and all its dependencies:

```
$ nix-store --verify-path $(nix-store --query --requisites $(which svn))
```

Name

`nix-store --verify` - check Nix database for consistency

Synopsis

`nix-store --verify [--check-contents] [--repair]`

Description

The operation `--verify` verifies the internal consistency of the Nix database, and the consistency between the Nix database and the Nix store. Any inconsistencies encountered are automatically repaired. Inconsistencies are generally the result of the Nix store or database being modified by non-Nix tools, or of bugs in Nix itself.

This operation has the following options:

- `--check-contents`

Checks that the contents of every valid store path has not been altered by computing a SHA-256 hash of the contents and comparing it with the hash stored in the Nix database at build time. Paths that have been modified are printed out. For large stores, `--check-contents` is obviously quite slow.

- `--repair`

If any valid path is missing from the store, or (if `--check-contents` is given) the contents of a valid path has been modified, then try to repair the path by redownloading it. See `nix-store --repair-path` for details.

Options

The following options are allowed for all `nix-store` operations, but may not always have an effect.

- `--add-root path`

Causes the result of a realisation (`--realise` and `--force-realise`) to be registered

as a root of the garbage collector. *path* will be created as a symlink to the resulting store path. In addition, a uniquely named symlink to *path* will be created in `/nix/var/nix/gcroots/auto/`. For instance,

```
$ nix-store --add-root /home/eelco/bla/result --realise ...  
  
$ ls -l /nix/var/nix/gcroots/auto  
lrwxrwxrwx    1 ... 2005-03-13 21:10 dn54lcypm8f8... -> /home/eelco  
/bla/result  
  
$ ls -l /home/eelco/bla/result  
lrwxrwxrwx    1 ... 2005-03-13 21:10 /home/eelco/bla/result -> /nix/store  
/1r11343n6qd4...-f-spot-0.0.10
```

Thus, when `/home/eelco/bla/result` is removed, the GC root in the `auto` directory becomes a dangling symlink and will be ignored by the collector.

Warning

Note that it is not possible to move or rename GC roots, since the symlink in the `auto` directory will still point to the old location.

If there are multiple results, then multiple symlinks will be created by sequentially numbering symlinks beyond the first one (e.g., `foo`, `foo-2`, `foo-3`, and so on).

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose` / `-v`

Increases the level of verbosity of diagnostic messages printed on standard error. For

each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- **0** “Errors only”

Only print messages explaining why the Nix invocation failed.

- **1** “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- **2** “Talkative”

Print more informational messages.

- **3** “Chatty”

Print even more informational messages.

- **4** “Debug”

Print debug information.

- **5** “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to **-v** / **--verbose**.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- **raw**

This is the raw format, as outputted by nix-build.

- **internal-json**

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- `--no-build-output / -Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- `--max-jobs / -j number`

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- `--cores`

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- `--max-silent-time`

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- `--timeout`

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- `--keep-going / -k`

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- `--keep-failed / -K`

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- `--fallback`

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- `--readonly-mode`

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- `--arg name value`

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`.

When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.  
  system ? builtins.currentSystem  
  ...  
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgnome --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- **--argstr** *name value*

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- **--attr** / **-A** *attrPath*

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression *e*, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- **--expr** / **-E**

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`,

`nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using [paths](#) enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in [impure](#) and [unrestricted](#) evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`

3. /nix/var/nix/profiles/per-user/root/channels

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **NIX_IGNORE_SYMLINK_STORE**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated

as if it was read from a Nix configuration file. Settings are separated by the newline character.

- [NIX_USER_CONF_FILES](#)

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- `XDG_CONFIG_HOME` (default `~/.config`)
- `XDG_STATE_HOME` (default `~/.local/state`)
- `XDG_CACHE_HOME` (default `~/.cache`)

Name

`nix-env` - manipulate or query Nix user environments

Synopsis

```
nix-env operation [options] [arguments...] [ --option name value] [ --arg name value]
[ --argstr name value] [{ --file | -f } path] [{ --profile | -p } path] [ --system-filter
system] [ --dry-run ]
```

Description

The command `nix-env` is used to manipulate Nix user environments. User environments are sets of software packages available to a user at some point in time. In other words, they are a synthesised view of the programs available in the Nix store. There may be many user environments: different users can have different environments, and individual users can switch between different environments.

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

- `--install`
- `--upgrade`
- `--uninstall`
- `--set`
- `--set-flag`
- `--query`
- `--switch-profile`
- `--list-generations`
- `--delete-generations`
- `--switch-generation`
- `--rollback`

These pages can be viewed offline:

- `man nix-env-<operation> .`

Example: `man nix-env-install`

- `nix-env --help --<operation>`

Example: `nix-env --help --install`

Selectors

Several commands, such as `nix-env --query` and `nix-env --install`, take a list of arguments that specify the packages on which to operate. These are extended regular expressions that must match the entire name of the package. (For details on regular expressions, see `regex(7)`.) The match is case-sensitive. The regular expression can optionally be followed by a dash and a version number; if omitted, any version of the package will match. Here are some examples:

- `firefox`

Matches the package name `firefox` and any version.

- `firefox-32.0`

Matches the package name `firefox` and version `32.0`.

- `gtk\\+`

Matches the package name `gtk+`. The `+` character must be escaped using a backslash to prevent it from being interpreted as a quantifier, and the backslash must be escaped in turn with another backslash to ensure that the shell passes it on.

- `.*`

Matches any package name. This is the default for most commands.

- `'.*zip.*'`

Matches any package name containing the string `zip`. Note the dots: `'*zip*' does not work, because in a regular expression, the character * is interpreted as a quantifier.`

- `'.*(firefox|chromium).*'`

Matches any package name containing the strings `firefox` or `chromium`.

Files

`nix-env` operates on the following files.

Default Nix expression

The source for the default [Nix expressions](#) used by `nix-env`:

- `~/.nix-defexpr`
- `$XDG_STATE_HOME/nix/defexpr` if `use-xdg-base-directories` is set to `true`.

It is loaded as follows:

- If the default expression is a file, it is loaded as a Nix expression.
- If the default expression is a directory containing a `default.nix` file, that `default.nix` file is loaded as a Nix expression.
- If the default expression is a directory without a `default.nix` file, then its contents (both files and subdirectories) are loaded as Nix expressions. The expressions are combined into a single attribute set, each expression under an attribute with the same name as the original file or subdirectory. Subdirectories without a `default.nix` file are traversed recursively in search of more Nix expressions, but the names of these intermediate directories are not added to the attribute paths of the default Nix expression.

Then, the resulting expression is interpreted like this:

- If the expression is an attribute set, it is used as the default Nix expression.
- If the expression is a function, an empty set is passed as argument and the return value is used as the default Nix expression.

For example, if the default expression contains two files, `foo.nix` and `bar.nix`, then the default Nix expression will be equivalent to

```
{  
  foo = import ~/.nix-defexpr/foo.nix;  
  bar = import ~/.nix-defexpr/bar.nix;  
}
```

The file `manifest.nix` is always ignored.

The command `nix-channel` places a symlink to the user's current [channels profile](#) in this directory. This makes all subscribed channels available as attributes in the default expression.

User channel link

A symlink that ensures that `nix-env` can find your channels:

- `~/.nix-defexpr/channels`
- `$XDG_STATE_HOME/defexpr/channels` if `use-xdg-base-directories` is set to `true`.

This symlink points to:

- `$XDG_STATE_HOME/profiles/channels` for regular users
- `$NIX_STATE_DIR/profiles/per-user/root/channels` for `root`

In a multi-user installation, you may also have `~/.nix-defexpr/channels_root`, which links to the channels of the root user. `nix-env`:
`..:/nix-env.md`

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
/profile -> profile-7-link
lrwxrwxrwx 1 alice users 51 Oct 28 16:18 /home/alice/.local/state/nix/profiles
/profile-5-link -> /nix/store/q69xad13ghpf7ir87h0b2gd28lafjj1j-profile
lrwxrwxrwx 1 alice users 51 Oct 29 13:20 /home/alice/.local/state/nix/profiles
/profile-6-link -> /nix/store/6vhpyasd7vwz7k3b0pndn7ifi5xr32dg-profile
lrwxrwxrwx 1 alice users 51 Nov 25 14:35 /home/alice/.local/state/nix/profiles
/profile-7-link -> /nix/store/mp0x6xnsg0b8qhsyw6riqvimai4gm677-profile
```

Each of these symlinks is a root for the Nix garbage collector.

The contents of the store path corresponding to each version of the profile is a tree of

symlinks to the files of the installed packages, e.g.

```
$ ll -R ~eelco/.local/state/nix/profiles/profile-7-link/
/home/eelco/.local/state/nix/profiles/profile-7-link/:
total 20
dr-xr-xr-x 2 root root 4096 Jan  1 1970 bin
-r--r--r-- 2 root root 1402 Jan  1 1970 manifest.nix
dr-xr-xr-x 4 root root 4096 Jan  1 1970 share

/home/eelco/.local/state/nix/profiles/profile-7-link/bin:
total 20
lrwxrwxrwx 5 root root 79 Jan  1 1970 chromium -> /nix/store
/ijm5k0zqisvkdwjk77mb9zb35xfi4m-chromium-86.0.4240.111/bin/chromium
lrwxrwxrwx 7 root root 87 Jan  1 1970 spotify -> /nix/store
/w9182874m1bl56sm3m5zjj36jhp3rn-spotify-1.1.26.501.gbe11e53b-15/bin/spotify
lrwxrwxrwx 3 root root 79 Jan  1 1970 zoom-us -> /nix/store
/wbhg2ga8f3h87s9h5k0slxk0m81m4cxl-zoom-us-5.3.469451.0927/bin/zoom-us

/home/eelco/.local/state/nix/profiles/profile-7-link/share/applications:
total 12
lrwxrwxrwx 4 root root 120 Jan  1 1970 chromium-browser.desktop -> /nix/store
/4cf803y4vzfm3gyk3vzhzb2327v0kl8a-chromium-unwrapped-86.0.4240.111/share
/applications/chromium-browser.desktop
lrwxrwxrwx 7 root root 110 Jan  1 1970 spotify.desktop -> /nix/store
/w9182874m1bl56sm3m5zjj36jhp3rn-spotify-1.1.26.501.gbe11e53b-15/share
/applications/spotify.desktop
lrwxrwxrwx 3 root root 107 Jan  1 1970 us.zoom.Zoom.desktop -> /nix/store
/wbhg2ga8f3h87s9h5k0slxk0m81m4cxl-zoom-us-5.3.469451.0927/share/applications
/us.zoom.Zoom.desktop

...
```

Each profile version contains a manifest file:

- `manifest.nix` used by `nix-env`.
- `manifest.json` used by `nix profile` (experimental).

User profile link

A symbolic link to the user's current profile:

- `~/.nix-profile`
- `$XDG_STATE_HOME/nix/profile` if `use-xdg-base-directories` is set to `true`.

By default, this symlink points to:

- `$XDG_STATE_HOME/nix/profiles/profile` for regular users
- `$NIX_STATE_DIR/profiles/per-user/root/profile` for `root`

The `PATH` environment variable should include `/bin` subdirectory of the profile link (e.g. `~/.nix-profile/bin`) for the user environment to be visible to the user. The `installer` sets this up by default, unless you enable `use-xdg-base-directories`.

Name

`nix-env --delete-generations` - delete profile generations

Synopsis

`nix-env --delete-generations generations`

Description

This operation deletes the specified generations of the current profile.

generations can be one of the following:

- `<number>...`:

A list of generation numbers, each one a separate command-line argument.

Delete exactly the profile generations given by their generation number. Deleting the current generation is not allowed.

- The special value `old`

Delete all generations except the current one.

WARNING

Older *and newer* generations will be deleted by this operation.

One might expect this to just delete older generations than the current one, but that is only true if the current generation is also the latest. Because one can roll back to a previous generation, it is possible to have generations newer than the current one. They will also be deleted.

-
- `<number>d`:

The last *number* days

Example: `30d`

Delete all generations created more than *number* days ago, except the most recent one of them. This allows rolling back to generations that were available within the specified period.

- `+<number>` :

The last *number* generations up to the present

Example: `+5`

Keep the last *number* generations, along with any newer than current.

Periodically deleting old generations is important to make garbage collection effective. This is because profiles are also garbage collection roots — any `store object` reachable from a profile is "alive" and ineligible for deletion.

Options

The following options are allowed for all `nix-env` operations, but may not always have an effect.

- `--file / -f path`

Specifies the Nix expression (designated below as the *active Nix expression*) used by the `--install`, `--upgrade`, and `--query --available` operations to obtain derivations.

The default is `~/.nix-defexpr`.

If the argument starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must include a single top-level directory containing at least a file named `default.nix`.

- `--profile / -p path`

Specifies the profile to be used by those operations that operate on a profile (designated below as the *active profile*). A profile is a sequence of user environments called *generations*, one of which is the *current generation*.

- `--dry-run`

For the `--install`, `--upgrade`, `--uninstall`, `--switch-generation`, `--delete-generations` and `--rollback` operations, this flag will cause `nix-env` to print what would be done if this flag had not been specified, without actually doing it.

`--dry-run` also prints out which paths will be `substituted` (i.e., downloaded) and which paths will be built from source (because no substitute is available).

- `--system-filter system`

By default, operations such as `--query --available` show derivations matching any platform. This option allows you to use derivations for the specified platform *system*.

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose / -v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- `--quiet`

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- `--log-format` *format*

This option can be used to change the output of the log format, with *format* being one of:

- raw

This is the raw format, as outputted by nix-build.

- internal-json

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg` -field) can change between releases.

- bar

Only display a progress bar during the builds.

- bar-with-logs

Display the raw logs, with the progress bar at the bottom.

- `--no-build-output` / `-Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in `Nixpkgs`, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- `--readonly-mode`

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- `--arg name value`

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a `default value` (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.
  system ? builtins.currentSystem
  ...
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgnome --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- `--argstr name value`

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- `--attr / -A attrPath`

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path* `attrPath` is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression `e`, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- `--expr / -E`

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option `name` to `value`. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Environment variables

- **NIX_PROFILE**

Location of the Nix profile. Defaults to the target of the symlink `~/.nix-profile`, if it exists, or `/nix/var/nix/profiles/default` otherwise.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- **IN_NIX_SHELL**

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- **NIX_PATH**

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **NIX_IGNORE_SYMLINK_STORE**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you're symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you're better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- **TMPDIR**

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- **NIX_REMOTE**

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [`NIX_SHOW_STATS`](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [`NIX_COUNT_CALLS`](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [`GC_INITIAL_HEAP_SIZE`](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [`XDG_CONFIG_HOME`](#) (default `~/.config`)
- [`XDG_STATE_HOME`](#) (default `~/.local/state`)
- [`XDG_CACHE_HOME`](#) (default `~/.cache`)

Examples

Delete explicit generation numbers

```
$ nix-env --delete-generations 3 4 8
```

Delete the generations numbered 3, 4, and 8, so long as the current active generation is not any of those.

Keep most-recent by count (number of generations)

```
$ nix-env --delete-generations +5
```

Suppose 30 is the current generation, and we currently have generations numbered 20 through 32.

Then this command will delete generations 20 through 25 ($\leq 30 - 5$), and keep generations 26 through 31 ($> 30 - 5$).

Keep most-recent by time (number of days)

```
$ nix-env --delete-generations 30d
```

This command will delete all generations older than 30 days, except for the generation that was active 30 days ago (if it currently exists).

Delete all older

```
$ nix-env --profile other_profile --delete-generations old
```

Name

`nix-env --install` - add packages to user environment

Synopsis

```
nix-env { --install | -i } args... [{ --prebuilt-only | -b }] [{ --attr | -A }] [ --from-expression ][ -E ][ --from-profile path][ --preserve-installed | -P ][ --remove-all | -r ]
```

Description

The install operation creates a new user environment, based on the current generation of the active profile, to which a set of store paths described by *args* is added. The arguments *args* map to store paths in a number of possible ways:

- By default, *args* is a set of derivation names denoting derivations in the active Nix expression. These are realised, and the resulting output paths are installed. Currently installed derivations with a name equal to the name of a derivation being added are removed unless the option `--preserve-installed` is specified.

If there are multiple derivations matching a name in *args* that have the same name (e.g., `gcc-3.3.6` and `gcc-4.1.1`), then the derivation with the highest *priority* is used. A derivation can define a priority by declaring the `meta.priority` attribute. This attribute should be a number, with a higher value denoting a lower priority. The default priority is `5`.

If there are multiple matching derivations with the same priority, then the derivation with the highest version will be installed.

You can force the installation of multiple derivations with the same name by being specific about the versions. For instance, `nix-env --install gcc-3.3.6 gcc-4.1.1` will install both version of GCC (and will probably cause a user environment conflict!).

- If `--attr` (`-A`) is specified, the arguments are *attribute paths* that select attributes from the top-level Nix expression. This is faster than using derivation names and unambiguous. To find out the attribute paths of available packages, use `nix-env --query --available --attr-path .`

- If `--from-profile` *path* is given, *args* is a set of names denoting installed store paths in the profile *path*. This is an easy way to copy user environment elements from one profile to another.
- If `--from-expression` is given, *args* are Nix [functions](#) that are called with the active Nix expression as their single argument. The derivations returned by those function calls are installed. This allows derivations to be specified in an unambiguous way, which is necessary if there are multiple derivations with the same name.
- If *args* are [store derivations](#), then these are [realised](#), and the resulting output paths are installed.
- If *args* are store paths that are not store derivations, then these are [realised](#) and installed.
- By default all outputs are installed for each derivation. That can be reduced by setting `meta.outputsToInstall`.

Flags

- `--prebuilt-only / -b`

Use only derivations for which a substitute is registered, i.e., there is a pre-built binary available that can be downloaded in lieu of building the derivation. Thus, no packages will be built from source.

- `--preserve-installed / -P`

Do not remove derivations with a name matching one of the derivations being installed. Usually, trying to have two versions of the same package installed in the same generation of a profile will lead to an error in building the generation, due to file name clashes between the two versions. However, this is not the case for all packages.

- `--remove-all / -r`

Remove all previously installed packages first. This is equivalent to running `nix-env --uninstall '.*'` first, except that everything happens in a single transaction.

Options

The following options are allowed for all `nix-env` operations, but may not always have an effect.

- **--file / -f path**

Specifies the Nix expression (designated below as the *active Nix expression*) used by the `--install`, `--upgrade`, and `--query --available` operations to obtain derivations. The default is `~/.nix-defexpr`.

If the argument starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must include a single top-level directory containing at least a file named `default.nix`.

- **--profile / -p path**

Specifies the profile to be used by those operations that operate on a profile (designated below as the *active profile*). A profile is a sequence of user environments called *generations*, one of which is the *current generation*.

- **--dry-run**

For the `--install`, `--upgrade`, `--uninstall`, `--switch-generation`, `--delete-generations` and `--rollback` operations, this flag will cause `nix-env` to print what *would* be done if this flag had not been specified, without actually doing it.

`--dry-run` also prints out which paths will be `substituted` (i.e., downloaded) and which paths will be built from source (because no substitute is available).

- **--system-filter system**

By default, operations such as `--query --available` show derivations matching any platform. This option allows you to use derivations for the specified platform *system*.

Common Options

Most Nix commands accept the following command-line options:

- **--help**

Prints out a summary of the command syntax and exits.

- **--version**

Prints out the Nix version number on standard output and exits.

- **--verbose / -v**

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any

diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- **raw**

This is the raw format, as outputted by nix-build.

- **internal-json**

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- `--no-build-output / -Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix /var/log/nix`.

- `--max-jobs / -j number`

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- `--cores`

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- `--max-silent-time`

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-`

`silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`.

When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every

argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.  
  system ? builtins.currentSystem  
  ...  
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgname`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgname --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- **--argstr name value**

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- **--attr / -A attrPath**

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression `e`, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- **--expr / -E**

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Environment variables

- `NIX_PROFILE`

Location of the Nix profile. Defaults to the target of the symlink `~/.nix-profile`, if it exists, or `/nix/var/nix/profiles/default` otherwise.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc`

`/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **`NIX_IGNORE_SYMLINK_STORE`**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **`NIX_STORE_DIR`**

Overrides the location of the Nix store (default `prefix/store`).

- **`NIX_DATA_DIR`**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **`NIX_LOG_DIR`**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **`NIX_STATE_DIR`**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- [NIX_CONF_DIR](#)

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- [NIX_CONFIG](#)

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- [NIX_USER_CONF_FILES](#)

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- `XDG_CONFIG_HOME` (default `~/.config`)
- `XDG_STATE_HOME` (default `~/.local/state`)
- `XDG_CACHE_HOME` (default `~/.cache`)

Examples

To install a package using a specific attribute path from the active Nix expression:

```
$ nix-env --install --attr gcc40mips
installing `gcc-4.0.2'
$ nix-env --install --attr xorg.xorgserver
installing `xorg-server-1.2.0'
```

To install a specific version of `gcc` using the derivation name:

```
$ nix-env --install gcc-3.3.2
installing `gcc-3.3.2'
uninstalling `gcc-3.1'
```

Using attribute path for selecting a package is preferred, as it is much faster and there will not be multiple matches.

Note the previously installed version is removed, since `--preserve-installed` was not specified.

To install an arbitrary version:

```
$ nix-env --install gcc
installing `gcc-3.3.2'
```

To install all derivations in the Nix expression `foo.nix`:

```
$ nix-env --file ~/foo.nix --install '.*' 
```

To copy the store path with symbolic name `gcc` from another profile:

```
$ nix-env --install --from-profile /nix/var/nix/profiles/foo gcc 
```

To install a specific [store derivation] (typically created by `nix-instantiate`):

```
$ nix-env --install /nix/store/fibjb1bfbpm5mrsxc4mh2d8n37sxh91i-gcc-3.4.3.drv 
```

To install a specific output path:

```
$ nix-env --install /nix/store/y3cgx0xj1p4iv9x0pnnmdhr8iyg741vk-gcc-3.4.3 
```

To install from a Nix expression specified on the command-line:

```
$ nix-env --file ./foo.nix --install --expr \  
'f: (f {system = "i686-linux";}).subversionWithJava' 
```

I.e., this evaluates to `(f: (f {system = "i686-linux";}).subversionWithJava)` (`import ./foo.nix`), thus selecting the `subversionWithJava` attribute from the set returned by calling the function defined in `./foo.nix`.

A dry-run tells you which paths will be downloaded or built from source:

```
$ nix-env --file '<nixpkgs>' --install --attr hello --dry-run  
(dry run; not doing anything)  
installing 'hello-2.10'  
this path will be fetched (0.04 MiB download, 0.19 MiB unpacked):  
/nix/store/wkhdf9jinag5750mqlax6z2zbwhqb76n-hello-2.10  
... 
```

To install Firefox from the latest revision in the Nixpkgs/NixOS 14.12 channel:

```
$ nix-env --file https://github.com/NixOS/nixpkgs/archive/nixos-14.12.tar.gz  
--install --attr firefox 
```

Name

`nix-env --list-generations` - list profile generations

Synopsis

`nix-env --list-generations`

Description

This operation print a list of all the currently existing generations for the active profile. These may be switched to using the `--switch-generation` operation. It also prints the creation date of the generation, and indicates the current generation.

Options

The following options are allowed for all `nix-env` operations, but may not always have an effect.

- `--file / -f path`

Specifies the Nix expression (designated below as the *active Nix expression*) used by the `--install`, `--upgrade`, and `--query --available` operations to obtain derivations.

The default is `~/.nix-defexpr`.

If the argument starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must include a single top-level directory containing at least a file named `default.nix`.

- `--profile / -p path`

Specifies the profile to be used by those operations that operate on a profile (designated below as the *active profile*). A profile is a sequence of user environments called *generations*, one of which is the *current generation*.

- `--dry-run`

For the `--install`, `--upgrade`, `--uninstall`, `--switch-generation`, `--delete-generations` and `--rollback` operations, this flag will cause `nix-env` to print what

would be done if this flag had not been specified, without actually doing it.

--dry-run also prints out which paths will be substituted (i.e., downloaded) and which paths will be built from source (because no substitute is available).

- `--system-filter system`

By default, operations such as --query --available show derivations matching any platform. This option allows you to use derivations for the specified platform *system*.

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose / -v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- `raw`

This is the raw format, as outputted by nix-build.

- `internal-json`

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg` -field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output** / `-Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that

the builder's standard output and error are always written to a log file in `prefix/nix` `/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is

printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in `Nixpkgs` is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.
  system ? builtins.currentSystem
  ...
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgname`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgname --arg system \"i686-freebsd\"`.

(Note that since the argument is a Nix string literal, you have to escape the quotes.)

- `--argstr name value`

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- `--attr / -A attrPath`

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path* `attrPath` is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression `e`, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- `--expr / -E`

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that

this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Environment variables

- `NIX_PROFILE`

Location of the Nix profile. Defaults to the target of the symlink `~/.nix-profile`, if it exists, or `/nix/var/nix/profiles/default` otherwise.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- `NIX_IGNORE_SYMLINK_STORE`

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes

“canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- [NIX_STORE_DIR](#)

Overrides the location of the Nix store (default `prefix/store`).

- [NIX_DATA_DIR](#)

Overrides the location of the Nix static data directory (default `prefix/share`).

- [NIX_LOG_DIR](#)

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- [NIX_STATE_DIR](#)

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- [NIX_CONF_DIR](#)

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- [NIX_CONFIG](#)

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- [NIX_USER_CONF_FILES](#)

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [XDG_CONFIG_HOME](#) (default `~/.config`)
- [XDG_STATE_HOME](#) (default `~/.local/state`)
- [XDG_CACHE_HOME](#) (default `~/.cache`)

Examples

```
$ nix-env --list-generations
95  2004-02-06 11:48:24
96  2004-02-06 11:49:01
97  2004-02-06 16:22:45
98  2004-02-06 16:24:33  (current)
```

Name

`nix-env --query` - display information about packages

Synopsis

```
nix-env { --query | -q } names... [ --installed | --available | -a ][{ --status | -s }] [{ --attr-path | -P }][ --no-name ][{ --compare-versions | -c }][ --system ][ --drv-path ][ --out-path ][ --description ][ --meta ][ --xml ][ --json ][{ --prebuilt-only | -b }][{ --attr | -A } attribute-path]
```

Description

The query operation displays information about either the store paths that are installed in the current generation of the active profile (`--installed`), or the derivations that are available for installation in the active Nix expression (`--available`). It only prints information about derivations whose symbolic name matches one of *names*.

The derivations are sorted by their `name` attributes.

Source selection

The following flags specify the set of things on which the query operates.

- `--installed`

The query operates on the store paths that are installed in the current generation of the active profile. This is the default.

- `--available; -a`

The query operates on the derivations that are available in the active Nix expression.

Queries

The following flags specify what information to display about the selected derivations.

Multiple flags may be specified, in which case the information is shown in the order given here. Note that the name of the derivation is shown unless `--no-name` is specified.

- `--xml`
Print the result in an XML representation suitable for automatic processing by other tools. The root element is called `items`, which contains a `item` element for each available or installed derivation. The fields discussed below are all stored in attributes of the `item` elements.
- `--json`
Print the result in a JSON representation suitable for automatic processing by other tools.
- `--prebuilt-only / -b`
Show only derivations for which a substitute is registered, i.e., there is a pre-built binary available that can be downloaded in lieu of building the derivation. Thus, this shows all packages that probably can be installed quickly.
- `--status ; -s`
Print the *status* of the derivation. The status consists of three characters. The first is `I` or `-`, indicating whether the derivation is currently installed in the current generation of the active profile. This is by definition the case for `--installed`, but not for `--available`. The second is `P` or `-`, indicating whether the derivation is present on the system. This indicates whether installation of an available derivation will require the derivation to be built. The third is `S` or `-`, indicating whether a substitute is available for the derivation.
- `--attr-path ; -P`
Print the *attribute path* of the derivation, which can be used to unambiguously select it using the `--attr` option available in commands that install derivations like `nix-env --install`. This option only works together with `--available`.
- `--no-name`
Suppress printing of the `name` attribute of each derivation.
- `--compare-versions / -c`
Compare installed versions to available versions, or vice versa (if `--available` is given). This is useful for quickly seeing whether upgrades for installed packages are available in a Nix expression. A column is added with the following meaning:
 - `< version`
A newer version of the package is available or installed.

- `= version`

At most the same version of the package is available or installed.

- `> version`

Only older versions of the package are available or installed.

- `- ?`

No version of the package is available or installed.

- **--system**

Print the `system` attribute of the derivation.

- **--drv-path**

Print the path of the [store derivation](#).

- **--out-path**

Print the output path of the derivation.

- **--description**

Print a short (one-line) description of the derivation, if available. The description is taken from the `meta.description` attribute of the derivation.

- **--meta**

Print all of the meta-attributes of the derivation. This option is only available with `--xml` or `--json`.

Options

The following options are allowed for all `nix-env` operations, but may not always have an effect.

- **--file / -f path**

Specifies the Nix expression (designated below as the *active Nix expression*) used by the `--install`, `--upgrade`, and `--query --available` operations to obtain derivations. The default is `~/.nix-defexpr`.

If the argument starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must include a single top-level directory containing at least a file named `default.nix`.

- **--profile / -p path**

Specifies the profile to be used by those operations that operate on a profile

(designated below as the *active profile*). A profile is a sequence of user environments called *generations*, one of which is the *current generation*.

- **--dry-run**

For the `--install`, `--upgrade`, `--uninstall`, `--switch-generation`, `--delete-generations` and `--rollback` operations, this flag will cause `nix-env` to print what would be done if this flag had not been specified, without actually doing it.

`--dry-run` also prints out which paths will be `substituted` (i.e., downloaded) and which paths will be built from source (because no substitute is available).

- **--system-filter system**

By default, operations such as `--query` `--available` show derivations matching any platform. This option allows you to use derivations for the specified platform *system*.

Common Options

Most Nix commands accept the following command-line options:

- **--help**

Prints out a summary of the command syntax and exits.

- **--version**

Prints out the Nix version number on standard output and exits.

- **--verbose / -v**

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- `raw`

This is the raw format, as outputted by nix-build.

- `internal-json`

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- **bar-with-logs**

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output / -Q**

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix` `/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an

input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.  
  system ? builtins.currentSystem  
  ...  
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgname`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgname --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- **--argstr** *name value*

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- **--attr** / **-A** *attrPath*

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression `e`, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- **--expr** / **-E**

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- **-I** *path*

Add an entry to the [Nix expression search path](#). This option may be given multiple

times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Environment variables

- `NIX_PROFILE`

Location of the Nix profile. Defaults to the target of the symlink `~/.nix-profile`, if it exists, or `/nix/var/nix/profiles/default` otherwise.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- [NIX_IGNORE_SYMLINK_STORE](#)

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- [NIX_STORE_DIR](#)

Overrides the location of the Nix store (default `prefix/store`).

- [NIX_DATA_DIR](#)

Overrides the location of the Nix static data directory (default `prefix/share`).

- [NIX_LOG_DIR](#)

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- [NIX_STATE_DIR](#)

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- [NIX_CONF_DIR](#)

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- [NIX_CONFIG](#)

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- [NIX_USER_CONF_FILES](#)

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- `XDG_CONFIG_HOME` (default `~/.config`)
- `XDG_STATE_HOME` (default `~/.local/state`)
- `XDG_CACHE_HOME` (default `~/.cache`)

Examples

To show installed packages:

```
$ nix-env --query
bison-1.875c
docbook-xml-4.2
firefox-1.0.4
MPlayer-1.0pre7
ORBit2-2.8.3
...
```

To show available packages:

```
$ nix-env --query --available
firefox-1.0.7
GConf-2.4.0.1
MPlayer-1.0pre7
ORBit2-2.8.3
...
```

To show the status of available packages:

```
$ nix-env --query --available --status
-P- firefox-1.0.7    (not installed but present)
--S GConf-2.4.0.1    (not present, but there is a substitute for fast
installation)
--S MPlayer-1.0pre3 (i.e., this is not the installed MPlayer, even though the
version is the same!)
IP- ORBit2-2.8.3    (installed and by definition present)
...
```

To show available packages in the Nix expression `foo.nix`:

```
$ nix-env --file ./foo.nix --query --available
foo-1.2.3
```

To compare installed versions to what's available:

```
$ nix-env --query --compare-versions  
...  
acrobat-reader-7.0 - ?      (package is not available at all)  
autoconf-2.59      = 2.59   (same version)  
firefox-1.0.4       < 1.0.7  (a more recent version is available)  
...
```

To show all packages with “ `zip` ” in the name:

```
$ nix-env --query --available '.*zip.*'  
bzip2-1.0.6  
gzip-1.6  
zip-3.0  
...
```

To show all packages with “ `firefox` ” or “ `chromium` ” in the name:

```
$ nix-env --query --available '.*(firefox|chromium).*'  
chromium-37.0.2062.94  
chromium-beta-38.0.2125.24  
firefox-32.0.3  
firefox-with-plugins-13.0.1  
...
```

To show all packages in the latest revision of the Nixpkgs repository:

```
$ nix-env --file https://github.com/NixOS/nixpkgs/archive/master.tar.gz --query  
--available
```

Name

`nix-env --rollback` - set user environment to previous generation

Synopsis

`nix-env --rollback`

Description

This operation switches to the “previous” generation of the active profile, that is, the highest numbered generation lower than the current generation, if it exists. It is just a convenience wrapper around `--list-generations` and `--switch-generation`.

Options

The following options are allowed for all `nix-env` operations, but may not always have an effect.

- `--file / -f path`

Specifies the Nix expression (designated below as the *active Nix expression*) used by the `--install`, `--upgrade`, and `--query --available` operations to obtain derivations.

The default is `~/.nix-defexpr`.

If the argument starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must include a single top-level directory containing at least a file named `default.nix`.

- `--profile / -p path`

Specifies the profile to be used by those operations that operate on a profile (designated below as the *active profile*). A profile is a sequence of user environments called *generations*, one of which is the *current generation*.

- `--dry-run`

For the `--install`, `--upgrade`, `--uninstall`, `--switch-generation`, `--delete-generations` and `--rollback` operations, this flag will cause `nix-env` to print what

would be done if this flag had not been specified, without actually doing it.

--dry-run also prints out which paths will be substituted (i.e., downloaded) and which paths will be built from source (because no substitute is available).

- `--system-filter system`

By default, operations such as --query --available show derivations matching any platform. This option allows you to use derivations for the specified platform *system*.

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose / -v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- `raw`

This is the raw format, as outputted by nix-build.

- `internal-json`

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg` -field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output** / `-Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that

the builder's standard output and error are always written to a log file in `prefix/nix` `/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is

printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in `Nixpkgs` is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.
  system ? builtins.currentSystem
  ...
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgname`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgname --arg system \"i686-freebsd\"`.

(Note that since the argument is a Nix string literal, you have to escape the quotes.)

- `--argstr name value`

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- `--attr / -A attrPath`

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression `e`, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- `--expr / -E`

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that

this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Environment variables

- `NIX_PROFILE`

Location of the Nix profile. Defaults to the target of the symlink `~/.nix-profile`, if it exists, or `/nix/var/nix/profiles/default` otherwise.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- `NIX_IGNORE_SYMLINK_STORE`

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes

“canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix  
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **`NIX_STORE_DIR`**

Overrides the location of the Nix store (default `prefix/store`).

- **`NIX_DATA_DIR`**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **`NIX_LOG_DIR`**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **`NIX_STATE_DIR`**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **`NIX_CONF_DIR`**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **`NIX_CONFIG`**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **`NIX_USER_CONF_FILES`**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [XDG_CONFIG_HOME](#) (default `~/.config`)
- [XDG_STATE_HOME](#) (default `~/.local/state`)
- [XDG_CACHE_HOME](#) (default `~/.cache`)

Examples

```
$ nix-env --rollback  
switching from generation 92 to 91  
  
$ nix-env --rollback  
error: no generation older than the current (91) exists
```

Name

`nix-env --set-flag` - modify meta attributes of installed packages

Synopsis

`nix-env --set-flag name value drvnames`

Description

The `--set-flag` operation allows meta attributes of installed packages to be modified. There are several attributes that can be usefully modified, because they affect the behaviour of `nix-env` or the user environment build script:

- `priority` can be changed to resolve filename clashes. The user environment build script uses the `meta.priority` attribute of derivations to resolve filename collisions between packages. Lower priority values denote a higher priority. For instance, the GCC wrapper package and the Binutils package in Nixpkgs both have a file `bin/ld`, so previously if you tried to install both you would get a collision. Now, on the other hand, the GCC wrapper declares a higher priority than Binutils, so the former's `bin/ld` is symlinked in the user environment.
- `keep` can be set to `true` to prevent the package from being upgraded or replaced. This is useful if you want to hang on to an older version of a package.
- `active` can be set to `false` to “disable” the package. That is, no symlinks will be generated to the files of the package, but it remains part of the profile (so it won't be garbage-collected). It can be set back to `true` to re-enable the package.

Options

The following options are allowed for all `nix-env` operations, but may not always have an effect.

- `--file / -f path`
Specifies the Nix expression (designated below as the *active Nix expression*) used by the

--install, --upgrade, and --query --available operations to obtain derivations.

The default is `~/.nix-defexpr`.

If the argument starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must include a single top-level directory containing at least a file named `default.nix`.

- `--profile / -p path`

Specifies the profile to be used by those operations that operate on a profile (designated below as the *active profile*). A profile is a sequence of user environments called *generations*, one of which is the *current generation*.

- `--dry-run`

For the --install, --upgrade, --uninstall, --switch-generation, --delete-generations and --rollback operations, this flag will cause `nix-env` to print what would be done if this flag had not been specified, without actually doing it.

--dry-run also prints out which paths will be substituted (i.e., downloaded) and which paths will be built from source (because no substitute is available).

- `--system-filter system`

By default, operations such as --query --available show derivations matching any platform. This option allows you to use derivations for the specified platform *system*.

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose / -v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- **0** “Errors only”

Only print messages explaining why the Nix invocation failed.

- **1** “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- **2** “Talkative”

Print more informational messages.

- **3** “Chatty”

Print even more informational messages.

- **4** “Debug”

Print debug information.

- **5** “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to **-v** / **--verbose**.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- **raw**

This is the raw format, as outputted by nix-build.

- **internal-json**

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- `--no-build-output / -Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix /var/log/nix`.

- `--max-jobs / -j number`

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- `--cores`

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- `--max-silent-time`

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-`

`silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`.

When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every

argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.  
  system ? builtins.currentSystem  
  ...  
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgname`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgname --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- **--argstr name value**

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- **--attr / -A attrPath**

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression `e`, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- **--expr / -E**

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I option`.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- [NIX_IGNORE_SYMLINK_STORE](#)

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- [NIX_STORE_DIR](#)

Overrides the location of the Nix store (default `prefix/store`).

- [NIX_DATA_DIR](#)

Overrides the location of the Nix static data directory (default `prefix/share`).

- [NIX_LOG_DIR](#)

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- [NIX_STATE_DIR](#)

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- [NIX_CONF_DIR](#)

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- [NIX_CONFIG](#)

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- [NIX_USER_CONF_FILES](#)

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- `XDG_CONFIG_HOME` (default `~/.config`)
- `XDG_STATE_HOME` (default `~/.local/state`)
- `XDG_CACHE_HOME` (default `~/.cache`)

Examples

To prevent the currently installed Firefox from being upgraded:

```
$ nix-env --set-flag keep true firefox
```

After this, `nix-env --upgrade` will ignore Firefox.

To disable the currently installed Firefox, then install a new Firefox while the old remains part of the profile:

```
$ nix-env --query
firefox-2.0.0.9 (the current one)

$ nix-env --preserve-installed --install firefox-2.0.0.11
installing `firefox-2.0.0.11'
building path(s) `/nix/store/myy0y59q3ig70dgq37jqwg1j0rsapzsl-user-environment'
collision between `/nix/store/...-firefox-2.0.0.11/bin/firefox'
and `/nix/store/...-firefox-2.0.0.9/bin/firefox'.
(i.e., can't have two active at the same time)

$ nix-env --set-flag active false firefox
setting flag on `firefox-2.0.0.9'

$ nix-env --preserve-installed --install firefox-2.0.0.11
installing `firefox-2.0.0.11'

$ nix-env --query
firefox-2.0.0.11 (the enabled one)
firefox-2.0.0.9 (the disabled one)
```

To make files from `binutils` take precedence over files from `gcc`:

```
$ nix-env --set-flag priority 5 binutils
$ nix-env --set-flag priority 10 gcc
```

Name

`nix-env --set` - set profile to contain a specified derivation

Synopsis

`nix-env --set drvname`

Description

The `--set` operation modifies the current generation of a profile so that it contains exactly the specified derivation, and nothing else.

Options

The following options are allowed for all `nix-env` operations, but may not always have an effect.

- `--file / -f path`

Specifies the Nix expression (designated below as the *active Nix expression*) used by the `--install`, `--upgrade`, and `--query --available` operations to obtain derivations. The default is `~/.nix-defexpr`.

If the argument starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must include a single top-level directory containing at least a file named `default.nix`.

- `--profile / -p path`

Specifies the profile to be used by those operations that operate on a profile (designated below as the *active profile*). A profile is a sequence of user environments called *generations*, one of which is the *current generation*.

- `--dry-run`

For the `--install`, `--upgrade`, `--uninstall`, `--switch-generation`, `--delete-generations` and `--rollback` operations, this flag will cause `nix-env` to print what would be done if this flag had not been specified, without actually doing it.

`--dry-run` also prints out which paths will be `substituted` (i.e., downloaded) and which paths will be built from source (because no substitute is available).

- `--system-filter system`

By default, operations such as `--query --available` show derivations matching any platform. This option allows you to use derivations for the specified platform *system*.

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose / -v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- `--quiet`

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- `--log-format` *format*

This option can be used to change the output of the log format, with *format* being one of:

- raw

This is the raw format, as outputted by nix-build.

- internal-json

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg` -field) can change between releases.

- bar

Only display a progress bar during the builds.

- bar-with-logs

Display the raw logs, with the progress bar at the bottom.

- `--no-build-output` / `-Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in `Nixpkgs`, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- `--readonly-mode`

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- `--arg name value`

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a `default value` (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.
  system ? builtins.currentSystem
  ...
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgnome --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- `--argstr name value`

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- `--attr / -A attrPath`

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path* `attrPath` is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression `e`, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- `--expr / -E`

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option `name` to `value`. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Environment variables

- **NIX_PROFILE**

Location of the Nix profile. Defaults to the target of the symlink `~/.nix-profile`, if it exists, or `/nix/var/nix/profiles/default` otherwise.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- **IN_NIX_SHELL**

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- **NIX_PATH**

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **NIX_IGNORE_SYMLINK_STORE**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you're symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you're better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- **TMPDIR**

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- **NIX_REMOTE**

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [`NIX_SHOW_STATS`](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [`NIX_COUNT_CALLS`](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [`GC_INITIAL_HEAP_SIZE`](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [`XDG_CONFIG_HOME`](#) (default `~/.config`)
- [`XDG_STATE_HOME`](#) (default `~/.local/state`)
- [`XDG_CACHE_HOME`](#) (default `~/.cache`)

Examples

The following updates a profile such that its current generation will contain just Firefox:

```
$ nix-env --profile /nix/var/nix/profiles/browser --set firefox
```

Name

`nix-env --switch-generation` - set user environment to given profile generation

Synopsis

`nix-env { --switch-generation | -G } generation`

Description

This operation makes generation number *generation* the current generation of the active profile. That is, if the `profile` is the path to the active profile, then the symlink `profile` is made to point to `profile-generation-link`, which is in turn a symlink to the actual user environment in the Nix store.

Switching will fail if the specified generation does not exist.

Options

The following options are allowed for all `nix-env` operations, but may not always have an effect.

- `--file / -f path`

Specifies the Nix expression (designated below as the *active Nix expression*) used by the `--install`, `--upgrade`, and `--query --available` operations to obtain derivations. The default is `~/.nix-defexpr`.

If the argument starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must include a single top-level directory containing at least a file named `default.nix`.

- `--profile / -p path`

Specifies the profile to be used by those operations that operate on a profile (designated below as the *active profile*). A profile is a sequence of user environments called *generations*, one of which is the *current generation*.

- `--dry-run`

For the `--install`, `--upgrade`, `--uninstall`, `--switch-generation`, `--delete-generations` and `--rollback` operations, this flag will cause `nix-env` to print what *would* be done if this flag had not been specified, without actually doing it.

`--dry-run` also prints out which paths will be **substituted** (i.e., downloaded) and which paths will be built from source (because no substitute is available).

- `--system-filter system`

By default, operations such as `--query --available` show derivations matching any platform. This option allows you to use derivations for the specified platform *system*.

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose / -v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- `raw`

This is the raw format, as outputted by nix-build.

- `internal-json`

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- `--no-build-output / -Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix /var/log/nix`.

- `--max-jobs / -j number`

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- `--cores`

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in `Nixpkgs`, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- `--max-silent-time`

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- `--timeout`

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- `--keep-going / -k`

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in `Nixpkgs` is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.
  system ? builtins.currentSystem
  ...
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgname`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgname --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- `--argstr name value`

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- `--attr / -A attrPath`

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression *e*, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- `--expr / -E`

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- [--repair](#)

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under [nix-store --repair-path](#).

Environment variables

- [NIX_PROFILE](#)

Location of the Nix profile. Defaults to the target of the symlink [~/.nix-profile](#), if it exists, or [/nix/var/nix/profiles/default](#) otherwise.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- [IN_NIX_SHELL](#)

Indicator that tells if the current environment was set up by [nix-shell](#). It can have the values [pure](#) or [impure](#).

- [NIX_PATH](#)

A colon-separated list of directories used to look up the location of Nix expressions using [paths](#) enclosed in angle brackets (i.e., [`<path>`](#)), e.g. [/home/eelco/Dev:/etc/nixos](#). It can be extended using the [-I option](#).

If [NIX_PATH](#) is not set at all, Nix will fall back to the following list in [impure](#) and [unrestricted](#) evaluation mode:

1. [\\$HOME/.nix-defexpr/channels](#)
2. [nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs](#)
3. [/nix/var/nix/profiles/per-user/root/channels](#)

If [NIX_PATH](#) is set to an empty string, resolving search paths will always fail. For example, attempting to use [`<nixpkgs>`](#) will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- [NIX_IGNORE_SYMLINK_STORE](#)

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- [NIX_STORE_DIR](#)

Overrides the location of the Nix store (default `prefix/store`).

- [NIX_DATA_DIR](#)

Overrides the location of the Nix static data directory (default `prefix/share`).

- [NIX_LOG_DIR](#)

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- [NIX_STATE_DIR](#)

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- [NIX_CONF_DIR](#)

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- [NIX_CONFIG](#)

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- [NIX_USER_CONF_FILES](#)

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [XDG_CONFIG_HOME](#) (default `~/.config`)
- [XDG_STATE_HOME](#) (default `~/.local/state`)
- [XDG_CACHE_HOME](#) (default `~/.cache`)

Examples

```
$ nix-env --switch-generation 42
switching from generation 50 to 42
```

Name

`nix-env --switch-profile` - set user environment to given profile

Synopsis

`nix-env { --switch-profile | -S } path`

Description

This operation makes *path* the current profile for the user. That is, the symlink `~/.nix-profile` is made to point to *path*.

Options

The following options are allowed for all `nix-env` operations, but may not always have an effect.

- `--file / -f path`

Specifies the Nix expression (designated below as the *active Nix expression*) used by the `--install`, `--upgrade`, and `--query --available` operations to obtain derivations. The default is `~/.nix-defexpr`.

If the argument starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must include a single top-level directory containing at least a file named `default.nix`.

- `--profile / -p path`

Specifies the profile to be used by those operations that operate on a profile (designated below as the *active profile*). A profile is a sequence of user environments called *generations*, one of which is the *current generation*.

- `--dry-run`

For the `--install`, `--upgrade`, `--uninstall`, `--switch-generation`, `--delete-generations` and `--rollback` operations, this flag will cause `nix-env` to print what would be done if this flag had not been specified, without actually doing it.

`--dry-run` also prints out which paths will be **substituted** (i.e., downloaded) and which paths will be built from source (because no substitute is available).

- `--system-filter system`

By default, operations such as `--query --available` show derivations matching any platform. This option allows you to use derivations for the specified platform *system*.

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose / -v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- `5` “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- `raw`

This is the raw format, as outputted by nix-build.

- `internal-json`

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output** / `-Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix`

`/var/log/nix .`

- `--max-jobs / -j number`

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- `--cores`

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- `--max-silent-time`

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- `--timeout`

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- `--keep-going / -k`

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- `--keep-failed / -K`

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg** *name value*

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.
  system ? builtins.currentSystem
  ...
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgnome --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- **--argstr** *name value*

This option is like **--arg**, only the value is not a Nix expression but a string. So instead of **--arg system \"i686-linux\"** (the outer quotes are to keep the shell happy) you can say **--argstr system i686-linux**.

- **--attr** / **-A** *attrPath*

Select an attribute from the top-level Nix expression being evaluated. (**nix-env**, **nix-instantiate**, **nix-build** and **nix-shell** only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression *e*, the attribute path **xorg.xorgserver** would cause the expression **e.xorg.xorgserver** to be used. See **nix-env --install** for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path **foo.3.bar** selects the **bar** attribute of the fourth element of the array in the **foo** attribute of the top-level expression.

- **--expr** / **-E**

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (**nix-instantiate**, **nix-build** and **nix-shell** only.)

For **nix-shell**, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the **nix-shell --packages** convenience flag instead.

- **-I** *path*

Add an entry to the **Nix expression search path**. This option may be given multiple times. Paths added through **-I** take precedence over **NIX_PATH**.

- **--option** *name value*

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- **--repair**

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under **nix-store**

```
--repair-path .
```

Environment variables

- **NIX_PROFILE**

Location of the Nix profile. Defaults to the target of the symlink `~/.nix-profile`, if it exists, or `/nix/var/nix/profiles/default` otherwise.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- **IN_NIX_SHELL**

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- **NIX_PATH**

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **NIX_IGNORE_SYMLINK_STORE**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different

results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you're not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you're symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you're better off using `bind` mount points, e.g.,

```
$ mkdir /nix  
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **`NIX_STORE_DIR`**

Overrides the location of the Nix store (default `prefix/store`).

- **`NIX_DATA_DIR`**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **`NIX_LOG_DIR`**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **`NIX_STATE_DIR`**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **`NIX_CONF_DIR`**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **`NIX_CONFIG`**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **`NIX_USER_CONF_FILES`**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- **`TMPDIR`**

Use the specified directory to store temporary files. In particular, this includes

temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [XDG_CONFIG_HOME](#) (default `~/.config`)
- [XDG_STATE_HOME](#) (default `~/.local/state`)
- [XDG_CACHE_HOME](#) (default `~/.cache`)

Examples

```
$ nix-env --switch-profile ~/my-profile
```

Name

`nix-env --uninstall` - remove packages from user environment

Synopsis

`nix-env { --uninstall | -e } drvnames...`

Description

The `uninstall` operation creates a new user environment, based on the current generation of the active profile, from which the store paths designated by the symbolic names *drvnames* are removed.

Options

The following options are allowed for all `nix-env` operations, but may not always have an effect.

- `--file / -f path`

Specifies the Nix expression (designated below as the *active Nix expression*) used by the `--install`, `--upgrade`, and `--query --available` operations to obtain derivations.

The default is `~/.nix-defexpr`.

If the argument starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must include a single top-level directory containing at least a file named `default.nix`.

- `--profile / -p path`

Specifies the profile to be used by those operations that operate on a profile (designated below as the *active profile*). A profile is a sequence of user environments called *generations*, one of which is the *current generation*.

- `--dry-run`

For the `--install`, `--upgrade`, `--uninstall`, `--switch-generation`, `--delete-generations` and `--rollback` operations, this flag will cause `nix-env` to print what

would be done if this flag had not been specified, without actually doing it.

--dry-run also prints out which paths will be substituted (i.e., downloaded) and which paths will be built from source (because no substitute is available).

- `--system-filter system`

By default, operations such as --query --available show derivations matching any platform. This option allows you to use derivations for the specified platform *system*.

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose / -v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- `raw`

This is the raw format, as outputted by nix-build.

- `internal-json`

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg` -field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output** / `-Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that

the builder's standard output and error are always written to a log file in `prefix/nix` `/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is

printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in `Nixpkgs` is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.
  system ? builtins.currentSystem
  ...
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgname`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgname --arg system \"i686-freebsd\"`.

(Note that since the argument is a Nix string literal, you have to escape the quotes.)

- `--argstr name value`

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- `--attr / -A attrPath`

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path* `attrPath` is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression `e`, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- `--expr / -E`

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that

this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Environment variables

- `NIX_PROFILE`

Location of the Nix profile. Defaults to the target of the symlink `~/.nix-profile`, if it exists, or `/nix/var/nix/profiles/default` otherwise.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- `NIX_IGNORE_SYMLINK_STORE`

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes

“canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix  
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **`NIX_STORE_DIR`**

Overrides the location of the Nix store (default `prefix/store`).

- **`NIX_DATA_DIR`**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **`NIX_LOG_DIR`**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **`NIX_STATE_DIR`**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **`NIX_CONF_DIR`**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **`NIX_CONFIG`**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **`NIX_USER_CONF_FILES`**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [XDG_CONFIG_HOME](#) (default `~/.config`)
- [XDG_STATE_HOME](#) (default `~/.local/state`)
- [XDG_CACHE_HOME](#) (default `~/.cache`)

Examples

```
$ nix-env --uninstall gcc  
$ nix-env --uninstall '.*' (remove everything)
```

Name

`nix-env --upgrade` - upgrade packages in user environment

Synopsis

```
nix-env { --upgrade | -u } args [ --lt | --leq | --eq | --always ] [{ --prebuilt-only | -b }] [{ --attr | -A }] [ --from-expression ][ -E ] [ --from-profile path ] [ --preserve-installed | -P ]
```

Description

The upgrade operation creates a new user environment, based on the current generation of the active profile, in which all store paths are replaced for which there are newer versions in the set of paths described by *args*. Paths for which there are no newer versions are left untouched; this is not an error. It is also not an error if an element of *args* matches no installed derivations.

For a description of how *args* is mapped to a set of store paths, see [--install](#). If *args* describes multiple store paths with the same symbolic name, only the one with the highest version is installed.

Flags

- [--lt](#)

Only upgrade a derivation to newer versions. This is the default.

- [--leq](#)

In addition to upgrading to newer versions, also “upgrade” to derivations that have the same version. Version are not a unique identification of a derivation, so there may be many derivations that have the same version. This flag may be useful to force “synchronisation” between the installed and available derivations.

- [--eq](#)

Only “upgrade” to derivations that have the same version. This may not seem very useful, but it actually is, e.g., when there is a new release of Nixpkgs and you want to

replace installed applications with the same versions built against newer dependencies (to reduce the number of dependencies floating around on your system).

- **--always**

In addition to upgrading to newer versions, also “upgrade” to derivations that have the same or a lower version. I.e., derivations may actually be downgraded depending on what is available in the active Nix expression.

- **--prebuilt-only / -b**

Use only derivations for which a substitute is registered, i.e., there is a pre-built binary available that can be downloaded in lieu of building the derivation. Thus, no packages will be built from source.

- **--preserve-installed / -P**

Do not remove derivations with a name matching one of the derivations being installed. Usually, trying to have two versions of the same package installed in the same generation of a profile will lead to an error in building the generation, due to file name clashes between the two versions. However, this is not the case for all packages.

Options

The following options are allowed for all `nix-env` operations, but may not always have an effect.

- **--file / -f path**

Specifies the Nix expression (designated below as the *active Nix expression*) used by the `--install`, `--upgrade`, and `--query --available` operations to obtain derivations.

The default is `~/.nix-defexpr`.

If the argument starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must include a single top-level directory containing at least a file named `default.nix`.

- **--profile / -p path**

Specifies the profile to be used by those operations that operate on a profile (designated below as the *active profile*). A profile is a sequence of user environments called *generations*, one of which is the *current generation*.

- **--dry-run**

For the `--install`, `--upgrade`, `--uninstall`, `--switch-generation`, `--delete-generations` and `--rollback` operations, this flag will cause `nix-env` to print what

would be done if this flag had not been specified, without actually doing it.

--dry-run also prints out which paths will be substituted (i.e., downloaded) and which paths will be built from source (because no substitute is available).

- `--system-filter system`

By default, operations such as --query --available show derivations matching any platform. This option allows you to use derivations for the specified platform *system*.

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose / -v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- `raw`

This is the raw format, as outputted by nix-build.

- `internal-json`

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg` -field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output** / `-Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that

the builder's standard output and error are always written to a log file in `prefix/nix` `/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is

printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in `Nixpkgs` is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.
  system ? builtins.currentSystem
  ...
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgname`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgname --arg system \"i686-freebsd\"`.

(Note that since the argument is a Nix string literal, you have to escape the quotes.)

- `--argstr name value`

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- `--attr / -A attrPath`

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression `e`, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- `--expr / -E`

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that

this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Environment variables

- `NIX_PROFILE`

Location of the Nix profile. Defaults to the target of the symlink `~/.nix-profile`, if it exists, or `/nix/var/nix/profiles/default` otherwise.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- `NIX_IGNORE_SYMLINK_STORE`

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes

“canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix  
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- `NIX_STORE_DIR`

Overrides the location of the Nix store (default `prefix/store`).

- `NIX_DATA_DIR`

Overrides the location of the Nix static data directory (default `prefix/share`).

- `NIX_LOG_DIR`

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- `NIX_STATE_DIR`

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- `NIX_CONF_DIR`

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- `NIX_CONFIG`

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- `NIX_USER_CONF_FILES`

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- [XDG_CONFIG_HOME](#) (default `~/.config`)
- [XDG_STATE_HOME](#) (default `~/.local/state`)
- [XDG_CACHE_HOME](#) (default `~/.cache`)

Examples

```
$ nix-env --upgrade --attr nixpkgs.gcc  
upgrading `gcc-3.3.1' to `gcc-3.4'
```

When there are no updates available, nothing will happen:

```
$ nix-env --upgrade --attr nixpkgs.pan
```

Using `-A` is preferred when possible, as it is faster and unambiguous but it is also possible to upgrade to a specific version by matching the derivation name:

```
$ nix-env --upgrade gcc-3.3.2 --always  
upgrading `gcc-3.4' to `gcc-3.3.2'
```

To try to upgrade everything (matching packages based on the part of the derivation name without version):

```
$ nix-env --upgrade  
upgrading `hello-2.1.2' to `hello-2.1.3'  
upgrading `mozilla-1.2' to `mozilla-1.4'
```

Versions

The upgrade operation determines whether a derivation `y` is an upgrade of a derivation `x` by looking at their respective `name` attributes. The names (e.g., `gcc-3.3.1`) are split into two parts: the package name (`gcc`), and the version (`3.3.1`). The version part starts after the first dash not followed by a letter. `y` is considered an upgrade of `x` if their package names match, and the version of `y` is higher than that of `x`.

The versions are compared by splitting them into contiguous components of numbers and letters. E.g., `3.3.1pre5` is split into `[3, 3, 1, "pre", 5]`. These lists are then compared lexicographically (from left to right). Corresponding components `a` and `b` are compared as follows. If they are both numbers, integer comparison is used. If `a` is an empty string and `b` is a number, `a` is considered less than `b`. The special string component `pre` (for *pre-release*) is considered to be less than other components. String components are considered less than number components. Otherwise, they are compared lexicographically (i.e., using case-sensitive string comparison).

This is illustrated by the following examples:

```
1.0 < 2.3
2.1 < 2.3
2.3 = 2.3
2.5 > 2.3
3.1 > 2.3
2.3.1 > 2.3
2.3.1 > 2.3a
2.3pre1 < 2.3
2.3pre3 < 2.3pre12
2.3a < 2.3c
2.3pre1 < 2.3c
2.3pre1 < 2.3q
```

Utilities

This section lists utilities that you can use when you work with Nix.

Name

`nix-channel` - manage Nix channels

Synopsis

```
nix-channel { --add url [name] | --remove name | --list | --update [names...] |  
--list-generations | --rollback [generation] }
```

Description

Channels are a mechanism for referencing remote Nix expressions and conveniently retrieving their latest version.

The moving parts of channels are:

- The official channels listed at <https://nixos.org/channels>
- The user-specific list of [subscribed channels](#)
- The [downloaded channel contents](#)
- The [Nix expression search path](#), set with the `-I` option or the `NIX_PATH` environment variable

Note

The state of a subscribed channel is external to the Nix expressions relying on it. This may limit reproducibility.

Dependencies on other Nix expressions can be declared explicitly with:

- `fetchurl`, `fetchTarball`, or `fetchGit` in Nix expressions
- the `-I` option in command line invocations

This command has the following operations:

- `--add url [name]`

Add a channel *name* located at *url* to the list of subscribed channels. If *name* is omitted, default to the last component of *url*, with the suffixes `-stable` or `-unstable` removed.

Note

`--add` does not automatically perform an update. Use `--update` explicitly.

A channel URL must point to a directory containing a file `nixexprs.tar.gz`. At the top level, that tarball must contain a single directory with a `default.nix` file that serves as the channel's entry point.

- `--remove name`

Remove the channel *name* from the list of subscribed channels.

- `--list`

Print the names and URLs of all subscribed channels on standard output.

- `--update [names...]`

Download the Nix expressions of subscribed channels and create a new generation. Update all channels if none is specified, and only those included in *names* otherwise.

- `--list-generations`

Prints a list of all the current existing generations for the channel profile.

Works the same way as

```
nix-env --profile /nix/var/nix/profiles/per-user/$USER/channels --list-generations
```

- `--rollback [generation]`

Revert channels to the state before the last call to `nix-channel --update`. Optionally, you can specify a specific channel *generation* number to restore.

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose` / `-v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- `--quiet`

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- `--log-format` *format*

This option can be used to change the output of the log format, with *format* being one of:

- `raw`

This is the raw format, as outputted by `nix-build`.

- `internal-json`

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- `--no-build-output / -Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- `--max-jobs / -j number`

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- `--cores`

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in `Nixpkgs`, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise.

The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- `--arg name value`

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a `default value` (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.  
  system ? builtins.currentSystem  
  ...  
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgnome --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- `--argstr name value`

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- `--attr / -A attrPath`

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression *e*, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- `--expr / -E`

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate` , `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I option`.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **`NIX_IGNORE_SYMLINK_STORE`**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **`NIX_STORE_DIR`**

Overrides the location of the Nix store (default `prefix/store`).

- **`NIX_DATA_DIR`**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **`NIX_LOG_DIR`**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **`NIX_STATE_DIR`**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **`NIX_CONF_DIR`**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- [NIX_CONFIG](#)

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- [NIX_USER_CONF_FILES](#)

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-`

`base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- `XDG_CONFIG_HOME` (default `~/.config`)
- `XDG_STATE_HOME` (default `~/.local/state`)
- `XDG_CACHE_HOME` (default `~/.cache`)

Files

`nix-channel` operates on the following files.

Channels

A directory containing symlinks to Nix channels, managed by `nix-channel`:

- `$XDG_STATE_HOME/nix/profiles/channels` for regular users
- `$NIX_STATE_DIR/profiles/per-user/root/channels` for `root`

`nix-channel` uses a [profile](#) to store channels. This profile contains symlinks to the contents of those channels.

Subscribed channels

The list of subscribed channels is stored in

- `~/.nix-channels`
- `$XDG_STATE_HOME/nix/channels` if `use-xdg-base-directories` is set to `true`

in the following format:

```
<url> <name>
...

```

Examples

Subscribe to the Nixpkgs channel and run `hello` from the GNU Hello package:

```
$ nix-channel --add https://nixos.org/channels/nixpkgs-unstable
$ nix-channel --list
nixpkgs https://nixos.org/channels/nixpkgs
$ nix-channel --update
$ nix-shell -p hello --run hello
hello
```

Revert channel updates using `--rollback`:

```
$ nix-instantiate --eval '<nixpkgs>' --attr lib.version
"22.11pre296212.530a53dc9"
$ nix-channel --rollback
switching from generation 483 to 482
$ nix-instantiate --eval '<nixpkgs>' --attr lib.version
"22.11pre281526.d0419badfad"
```

Remove a channel:

```
$ nix-channel --remove nixpkgs
$ nix-channel --list
```

Name

`nix-collect-garbage` - delete unreachable `store objects`

Synopsis

```
nix-collect-garbage [ --delete-old ] [ -d ] [ --delete-older-than period ] [ --max-freed bytes ] [ --dry-run ]
```

Description

The command `nix-collect-garbage` is mostly an alias of `nix-store --gc`. That is, it deletes all unreachable `store objects` in the Nix store to clean up your system.

However, it provides two additional options, `--delete-old` and `--delete-older-than`, which also delete old `profiles`, allowing potentially more `store objects` to be deleted because profiles are also garbage collection roots. These options are the equivalent of running `nix-env --delete-generations` with various augments on multiple profiles, prior to running `nix-collect-garbage` (or just `nix-store --gc`) without any flags.

Note

Deleting previous configurations makes rollbacks to them impossible.

These flags should be used with care, because they potentially delete generations of profiles used by other users on the system.

Locations searched for profiles

`nix-collect-garbage` cannot know about all profiles; that information doesn't exist. Instead, it looks in a few locations, and acts on all profiles it finds there:

1. The default profile locations as specified in the `profiles` section of the manual.

-
2. **NOTE**

Not stable; subject to change

Do not rely on this functionality; it just exists for migration purposes and is may change in the future. These deprecated paths remain a private implementation detail of Nix.

`$NIX_STATE_DIR/profiles` and `$NIX_STATE_DIR/profiles/per-user`.

With the exception of `$NIX_STATE_DIR/profiles/per-user/root` and `$NIX_STATE_DIR/profiles/default`, these directories are no longer used by other commands. `nix-collect-garbage` looks there anyways in order to clean up profiles from older versions of Nix.

Options

These options are for deleting old `profiles` prior to deleting unreachable `store objects`.

- `--delete-old` / `-d`

Delete all old generations of profiles.

This is the equivalent of invoking `nix-env --delete-generations old` on each found profile.

- `--delete-older-than` *period*

Delete all generations of profiles older than the specified amount (except for the generations that were active at that point in time). *period* is a value such as `30d`, which would mean 30 days.

This is the equivalent of invoking `nix-env --delete-generations <period>` on each found profile. See the documentation of that command for additional information about the *period* argument.

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- **--version**

Prints out the Nix version number on standard output and exits.

- **--verbose / -v**

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to **-v / --verbose**.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- `raw`

This is the raw format, as outputted by `nix-build`.

- `internal-json`

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- `--no-build-output / -Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix /var/log/nix`.

- `--max-jobs / -j number`

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- `--cores`

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in `Nixpkgs`, if the `derivation` attribute

`enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- `--arg name value`

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a `default value` (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named `name`, it will call it with value `value`.

For instance, the top-level `default.nix` in `Nixpkgs` is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.  
  system ? builtins.currentSystem  
  ...  
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgnome --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- `--argstr name value`

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- `--attr / -A attrPath`

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression `e`, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array

in the `foo` attribute of the top-level expression.

- `--expr / -E`

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I option`.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **`NIX_IGNORE_SYMLINK_STORE`**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **`NIX_STORE_DIR`**

Overrides the location of the Nix store (default `prefix/store`).

- **`NIX_DATA_DIR`**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **`NIX_LOG_DIR`**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **`NIX_STATE_DIR`**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **`NIX_CONF_DIR`**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- **NIX_USER_CONF_FILES**

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- **TMPDIR**

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- **NIX_REMOTE**

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- **NIX_SHOW_STATS**

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- **NIX_COUNT_CALLS**

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- **GC_INITIAL_HEAP_SIZE**

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- `XDG_CONFIG_HOME` (default `~/.config`)
- `XDG_STATE_HOME` (default `~/.local/state`)
- `XDG_CACHE_HOME` (default `~/.cache`)

Example

To delete from the Nix store everything that is not used by the current generations of each profile, do

```
$ nix-collect-garbage -d
```

Name

`nix-copy-closure` - copy a closure to or from a remote machine via SSH

Synopsis

```
nix-copy-closure [ --to | --from ] [ --gzip ] [ --include-outputs ] [ --use-substitutes |  
-s ] [ -v ] user@machine paths
```

Description

`nix-copy-closure` gives you an easy and efficient way to exchange software between machines. Given one or more Nix store *paths* on the local machine, `nix-copy-closure` computes the closure of those paths (i.e. all their dependencies in the Nix store), and copies all paths in the closure to the remote machine via the `ssh` (Secure Shell) command. With the `--from` option, the direction is reversed: the closure of *paths* on a remote machine is copied to the Nix store on the local machine.

This command is efficient because it only sends the store paths that are missing on the target machine.

Since `nix-copy-closure` calls `ssh`, you may be asked to type in the appropriate password or passphrase. In fact, you may be asked *twice* because `nix-copy-closure` currently connects twice to the remote machine, first to get the set of paths missing on the target machine, and second to send the dump of those paths. When using public key authentication, you can avoid typing the passphrase with `ssh-agent`.

Options

- `--to`

Copy the closure of *paths* from the local Nix store to the Nix store on *machine*. This is the default.

- `--from`

Copy the closure of *paths* from the Nix store on *machine* to the local Nix store.

- **--gzip**
Enable compression of the SSH connection.
- **--include-outputs**
Also copy the outputs of [store derivations](#)s included in the closure.
- **--use-substitutes / -s**
Attempt to download missing paths on the target machine using Nix's substitute mechanism. Any paths that cannot be substituted on the target are still copied normally from the source. This is useful, for instance, if the connection between the source and target machine is slow, but the connection between the target machine and [nixos.org](#) (the default binary cache server) is fast.
- **-v**
Show verbose output.

Common Options

Most Nix commands accept the following command-line options:

- **--help**
Prints out a summary of the command syntax and exits.
- **--version**
Prints out the Nix version number on standard output and exits.
- **--verbose / -v**
Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- 0 “Errors only”

Only print messages explaining why the Nix invocation failed.

- 1 “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- 2 “Talkative”

Print more informational messages.

- 3 “Chatty”

Print even more informational messages.

- 4 “Debug”

Print debug information.

- 5 “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to `-v` / `--verbose`.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- `raw`

This is the raw format, as outputted by nix-build.

- `internal-json`

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- **bar-with-logs**

Display the raw logs, with the progress bar at the bottom.

- **--no-build-output / -Q**

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix` `/var/log/nix`.

- **--max-jobs / -j number**

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- **--cores**

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- **--max-silent-time**

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- **--timeout**

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- **--keep-going / -k**

Keep going in case of failed builds, to the greatest extent possible. That is, if building an

input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- **--keep-failed / -K**

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- **--fallback**

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- **--readonly-mode**

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- **--arg name value**

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`. When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.  
  system ? builtins.currentSystem  
  ...  
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgname`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgname --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- **--argstr** *name value*

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- **--attr** / **-A** *attrPath*

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression `e`, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- **--expr** / **-E**

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`, `nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- **-I** *path*

Add an entry to the [Nix expression search path](#). This option may be given multiple

times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Environment variables

- `NIX_SSHOPTS`

Additional options to be passed to `ssh` on the command line.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in `impure` and `unrestricted` evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`
3. `/nix/var/nix/profiles/per-user/root/channels`

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For

example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **NIX_IGNORE_SYMLINK_STORE**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated as if it was read from a Nix configuration file. Settings are separated by the newline character.

- [NIX_USER_CONF_FILES](#)

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- `XDG_CONFIG_HOME` (default `~/.config`)
- `XDG_STATE_HOME` (default `~/.local/state`)
- `XDG_CACHE_HOME` (default `~/.cache`)

Examples

Copy Firefox with all its dependencies to a remote machine:

```
$ nix-copy-closure --to alice@itchy.labs $(type -tP firefox)
```

Copy Subversion from a remote machine and then install it into a user environment:

```
$ nix-copy-closure --from alice@itchy.labs \
  /nix/store/0dj0503hjxy5mbwlafv1rsbdiyx1gkdy-subversion-1.4.4
$ nix-env --install /nix/store/0dj0503hjxy5mbwlafv1rsbdiyx1gkdy-subversion-
  1.4.4
```

Name

nix-daemon - Nix multi-user support daemon

Synopsis

nix-daemon

Description

The Nix daemon is necessary in multi-user Nix installations. It runs build tasks and other operations on the Nix store on behalf of unprivileged users.

Name

`nix-hash` - compute the cryptographic hash of a path

Synopsis

```
nix-hash [ --flat ] [ --base32 ] [ --truncate ] [ --type hashAlgo ] path...
```

```
nix-hash [ --to-base16 | --to-base32 | --to-base64 | --to-sri ] [ --type hashAlgo ] hash...
```

Description

The command `nix-hash` computes the cryptographic hash of the contents of each *path* and prints it on standard output. By default, it computes an MD5 hash, but other hash algorithms are available as well. The hash is printed in hexadecimal. To generate the same hash as `nix-prefetch-url` you have to specify multiple arguments, see below for an example.

The hash is computed over a *serialisation* of each path: a dump of the file system tree rooted at the path. This allows directories and symlinks to be hashed as well as regular files. The dump is in the *NAR format* produced by `nix-store --dump`. Thus, `nix-hash path` yields the same cryptographic hash as `nix-store --dump path | md5sum`.

Options

- `--flat`

Print the cryptographic hash of the contents of each regular file *path*. That is, do not compute the hash over the dump of *path*. The result is identical to that produced by the GNU commands `md5sum` and `sha1sum`.

- `--base16`

Print the hash in a hexadecimal representation (default).

- `--base32`

Print the hash in a base-32 representation rather than hexadecimal. This base-32 representation is more compact and can be used in Nix expressions (such as in calls to

- `fetchurl`).
- **--base64**
Similar to --base32, but print the hash in a base-64 representation, which is more compact than the base-32 one.
 - **--sri**
Print the hash in SRI format with base-64 encoding. The type of hash algorithm will be prepended to the hash string, followed by a hyphen (-) and the base-64 hash body.
 - **--truncate**
Truncate hashes longer than 160 bits (such as SHA-256) to 160 bits.
 - **--type *hashAlgo***
Use the specified cryptographic hash algorithm, which can be one of `md5`, `sha1`, `sha256`, and `sha512`.
 - **--to-base16**
Don't hash anything, but convert the base-32 hash representation *hash* to hexadecimal.
 - **--to-base32**
Don't hash anything, but convert the hexadecimal hash representation *hash* to base-32.
 - **--to-base64**
Don't hash anything, but convert the hexadecimal hash representation *hash* to base-64.
 - **--to-sri**
Don't hash anything, but convert the hexadecimal hash representation *hash* to SRI.

Examples

Computing the same hash as `nix-prefetch-url`:

```
$ nix-prefetch-url file://<(echo test)  
1kgqb6fc1ns49861dwk9rzb6xnfkxbpws74mxnx01z9qyv1pjpj  
$ nix-hash --type sha256 --flat --base32 <(echo test)  
1kgqb6fc1ns49861dwk9rzb6xnfkxbpws74mxnx01z9qyv1pjpj
```

Computing hashes:

```
$ mkdir test
$ echo "hello" > test/world

$ nix-hash test/ (MD5 hash; default)
8179d3caeff1869b5ba1744e5a245c04

$ nix-store --dump test/ | md5sum (for comparison)
8179d3caeff1869b5ba1744e5a245c04 -

$ nix-hash --type sha1 test/
e4fd8ba5f7bbeaea5ace89fe10255536cd60dab6

$ nix-hash --type sha1 --base16 test/
e4fd8ba5f7bbeaea5ace89fe10255536cd60dab6

$ nix-hash --type sha1 --base32 test/
nvd61k9nalji1zl9rrdfmsmvyjqpzg4

$ nix-hash --type sha1 --base64 test/
5P2Lpfe76upazon+ECVVNs1g2rY=

$ nix-hash --type sha1 --sri test/
sha1-5P2Lpfe76upazon+ECVVNs1g2rY=

$ nix-hash --type sha256 --flat test/
error: reading file `test/': Is a directory

$ nix-hash --type sha256 --flat test/world
5891b5b522d5df086d0ff0b110fdb9d21bb4fc7163af34d08286a2e846f6be03
```

Converting between hexadecimal, base-32, base-64, and SRI:

```
$ nix-hash --type sha1 --to-base32 e4fd8ba5f7bbeaea5ace89fe10255536cd60dab6
nvd61k9nalji1zl9rrdfmsmvyjqpzg4

$ nix-hash --type sha1 --to-base16 nvd61k9nalji1zl9rrdfmsmvyjqpzg4
e4fd8ba5f7bbeaea5ace89fe10255536cd60dab6

$ nix-hash --type sha1 --to-base64 e4fd8ba5f7bbeaea5ace89fe10255536cd60dab6
5P2Lpfe76upazon+ECVVNs1g2rY=

$ nix-hash --type sha1 --to-sri nvd61k9nalji1zl9rrdfmsmvyjqpzg4
sha1-5P2Lpfe76upazon+ECVVNs1g2rY=

$ nix-hash --to-base16 sha1-5P2Lpfe76upazon+ECVVNs1g2rY=
e4fd8ba5f7bbeaea5ace89fe10255536cd60dab6
```

Name

`nix-instantiate` - instantiate store derivations from Nix expressions

Synopsis

```
nix-instantiate [ --parse | --eval [ --strict ] [ --json ] [ --xml ] ] [ --read-write-mode ] [ --arg name value ] [ { --attr | -A } attrPath ] [ --add-root path ] [ --expr | -E ] files...
```

```
nix-instantiate --find-file files...
```

Description

The command `nix-instantiate` produces [store derivations](#) from (high-level) Nix expressions. It evaluates the Nix expressions in each of *files* (which defaults to `./default.nix`). Each top-level expression should evaluate to a derivation, a list of derivations, or a set of derivations. The paths of the resulting store derivations are printed on standard output.

If *files* is the character `-`, then a Nix expression will be read from standard input.

Options

- `--add-root` *path*

See the [corresponding option](#) in `nix-store`.

- `--parse`

Just parse the input files, and print their abstract syntax trees on standard output in ATerm format.

- `--eval`

Just parse and evaluate the input files, and print the resulting values on standard output. No instantiation of store derivations takes place.

- `--find-file`

Look up the given files in Nix's search path (as specified by the `NIX_PATH` environment variable). If found, print the corresponding absolute paths on standard output. For

instance, if `NIX_PATH` is `nixpkgs=/home/alice/nixpkgs`, then `nix-instantiate --find-file nixpkgs/default.nix` will print `/home/alice/nixpkgs/default.nix`.

- `--strict`

When used with `--eval`, recursively evaluate list elements and attributes. Normally, such sub-expressions are left unevaluated (since the Nix language is lazy).

Warning

This option can cause non-termination, because lazy data structures can be infinitely large.

- `--json`

When used with `--eval`, print the resulting value as an JSON representation of the abstract syntax tree rather than as an ATerm.

- `--xml`

When used with `--eval`, print the resulting value as an XML representation of the abstract syntax tree rather than as an ATerm. The schema is the same as that used by the `toXML` built-in.

- `--read-write-mode`

When used with `--eval`, perform evaluation in read/write mode so nix language features that require it will still work (at the cost of needing to do instantiation of every evaluated derivation). If this option is not enabled, there may be uninstantiated store paths in the final output.

Common Options

Most Nix commands accept the following command-line options:

- `--help`

Prints out a summary of the command syntax and exits.

- `--version`

Prints out the Nix version number on standard output and exits.

- `--verbose / -v`

Increases the level of verbosity of diagnostic messages printed on standard error. For each Nix operation, the information printed on standard output is well-defined; any diagnostic information is printed on standard error, never on standard output.

This option may be specified repeatedly. Currently, the following verbosity levels exist:

- **0** “Errors only”

Only print messages explaining why the Nix invocation failed.

- **1** “Informational”

Print *useful* messages about what Nix is doing. This is the default.

- **2** “Talkative”

Print more informational messages.

- **3** “Chatty”

Print even more informational messages.

- **4** “Debug”

Print debug information.

- **5** “Vomit”

Print vast amounts of debug information.

- **--quiet**

Decreases the level of verbosity of diagnostic messages printed on standard error. This is the inverse option to **-v** / **--verbose**.

This option may be specified repeatedly. See the previous verbosity levels list.

- **--log-format** *format*

This option can be used to change the output of the log format, with *format* being one of:

- **raw**

This is the raw format, as outputted by nix-build.

- **internal-json**

Outputs the logs in a structured manner.

Warning

While the schema itself is relatively stable, the format of the error-messages (namely of the `msg`-field) can change between releases.

- `bar`

Only display a progress bar during the builds.

- `bar-with-logs`

Display the raw logs, with the progress bar at the bottom.

- `--no-build-output / -Q`

By default, output written by builders to standard output and standard error is echoed to the Nix command's standard error. This option suppresses this behaviour. Note that the builder's standard output and error are always written to a log file in `prefix/nix/var/log/nix`.

- `--max-jobs / -j number`

Sets the maximum number of build jobs that Nix will perform in parallel to the specified number. Specify `auto` to use the number of CPUs in the system. The default is specified by the `max-jobs` configuration setting, which itself defaults to `1`. A higher value is useful on SMP systems or to exploit I/O latency.

Setting it to `0` disallows building on the local machine, which is useful when you want builds to happen only on remote builders.

- `--cores`

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It defaults to the value of the `cores` configuration setting, if set, or `1` otherwise. The value `0` means that the builder should use all available CPU cores in the system.

- `--max-silent-time`

Sets the maximum number of seconds that a builder can go without producing any data on standard output or standard error. The default is specified by the `max-silent-time` configuration setting. `0` means no time-out.

- `--timeout`

Sets the maximum number of seconds that a builder can run. The default is specified by the `timeout` configuration setting. `0` means no timeout.

- `--keep-going / -k`

Keep going in case of failed builds, to the greatest extent possible. That is, if building an input of some derivation fails, Nix will still build the other inputs, but not the derivation itself. Without this option, Nix stops if any build fails (except for builds of substitutes), possibly killing builds in progress (in case of parallel or distributed builds).

- `--keep-failed / -K`

Specifies that in case of a build failure, the temporary directory (usually in `/tmp`) in which the build takes place should not be deleted. The path of the build directory is printed as an informational message.

- `--fallback`

Whenever Nix attempts to build a derivation for which substitutes are known for each output path, but realising the output paths through the substitutes fails, fall back on building the derivation.

The most common scenario in which this is useful is when we have registered substitutes in order to perform binary distribution from, say, a network repository. If the repository is down, the realisation of the derivation will fail. When this option is specified, Nix will build the derivation instead. Thus, installation from binaries falls back on installation from source. This option is not the default since it is generally not desirable for a transient failure in obtaining the substitutes to lead to a full build from source (with the related consumption of resources).

- `--readonly-mode`

When this option is used, no attempt is made to open the Nix database. Most Nix operations do need database access, so those operations will fail.

- `--arg name value`

This option is accepted by `nix-env`, `nix-instantiate`, `nix-shell` and `nix-build`.

When evaluating Nix expressions, the expression evaluator will automatically try to call functions that it encounters. It can automatically call functions for which every argument has a **default value** (e.g., `{ argName ? defaultValue }: ...`).

With `--arg`, you can also call functions that have arguments without a default value (or override a default value). That is, if the evaluator encounters a function with an argument named *name*, it will call it with value *value*.

For instance, the top-level `default.nix` in Nixpkgs is actually a function:

```
{ # The system (e.g., `i686-linux') for which to build the packages.  
  system ? builtins.currentSystem  
  ...  
}: ...
```

So if you call this Nix expression (e.g., when you do `nix-env --install --attr pkgnome`), the function will be called automatically using the value `builtins.currentSystem` for the `system` argument. You can override this using `--arg`, e.g., `nix-env --install --attr pkgnome --arg system \"i686-freebsd\"`. (Note that since the argument is a Nix string literal, you have to escape the quotes.)

- **--argstr** *name value*

This option is like `--arg`, only the value is not a Nix expression but a string. So instead of `--arg system \"i686-linux\"` (the outer quotes are to keep the shell happy) you can say `--argstr system i686-linux`.

- **--attr** / **-A** *attrPath*

Select an attribute from the top-level Nix expression being evaluated. (`nix-env`, `nix-instantiate`, `nix-build` and `nix-shell` only.) The *attribute path attrPath* is a sequence of attribute names separated by dots. For instance, given a top-level Nix expression *e*, the attribute path `xorg.xorgserver` would cause the expression `e.xorg.xorgserver` to be used. See `nix-env --install` for some concrete examples.

In addition to attribute names, you can also specify array indices. For instance, the attribute path `foo.3.bar` selects the `bar` attribute of the fourth element of the array in the `foo` attribute of the top-level expression.

- **--expr** / **-E**

Interpret the command line arguments as a list of Nix expressions to be parsed and evaluated, rather than as a list of file names of Nix expressions. (`nix-instantiate`,

`nix-build` and `nix-shell` only.)

For `nix-shell`, this option is commonly used to give you a shell in which you can build the packages returned by the expression. If you want to get a shell which contain the *built* packages ready for use, give your expression to the `nix-shell --packages` convenience flag instead.

- `-I path`

Add an entry to the [Nix expression search path](#). This option may be given multiple times. Paths added through `-I` take precedence over `NIX_PATH`.

- `--option name value`

Set the Nix configuration option *name* to *value*. This overrides settings in the Nix configuration file (see `nix.conf5`).

- `--repair`

Fix corrupted or missing store paths by redownloading or rebuilding them. Note that this is slow because it requires computing a cryptographic hash of the contents of every path in the closure of the build. Also note the warning under `nix-store --repair-path`.

Common Environment Variables

Most Nix commands interpret the following environment variables:

- `IN_NIX_SHELL`

Indicator that tells if the current environment was set up by `nix-shell`. It can have the values `pure` or `impure`.

- `NIX_PATH`

A colon-separated list of directories used to look up the location of Nix expressions using [paths](#) enclosed in angle brackets (i.e., `<path>`), e.g. `/home/eelco/Dev:/etc/nixos`. It can be extended using the `-I` option.

If `NIX_PATH` is not set at all, Nix will fall back to the following list in [impure](#) and [unrestricted](#) evaluation mode:

1. `$HOME/.nix-defexpr/channels`
2. `nixpkgs=/nix/var/nix/profiles/per-user/root/channels/nixpkgs`

3. /nix/var/nix/profiles/per-user/root/channels

If `NIX_PATH` is set to an empty string, resolving search paths will always fail. For example, attempting to use `<nixpkgs>` will produce:

```
error: file 'nixpkgs' was not found in the Nix search path
```

- **NIX_IGNORE_SYMLINK_STORE**

Normally, the Nix store directory (typically `/nix/store`) is not allowed to contain any symlink components. This is to prevent “impure” builds. Builders sometimes “canonicalise” paths by resolving all symlink components. Thus, builds on different machines (with `/nix/store` resolving to different locations) could yield different results. This is generally not a problem, except when builds are deployed to machines where `/nix/store` resolves differently. If you are sure that you’re not going to do that, you can set `NIX_IGNORE_SYMLINK_STORE` to `1`.

Note that if you’re symlinking the Nix store so that you can put it on another file system than the root file system, on Linux you’re better off using `bind` mount points, e.g.,

```
$ mkdir /nix
$ mount -o bind /mnt/otherdisk/nix /nix
```

Consult the `mount 8` manual page for details.

- **NIX_STORE_DIR**

Overrides the location of the Nix store (default `prefix/store`).

- **NIX_DATA_DIR**

Overrides the location of the Nix static data directory (default `prefix/share`).

- **NIX_LOG_DIR**

Overrides the location of the Nix log directory (default `prefix/var/log/nix`).

- **NIX_STATE_DIR**

Overrides the location of the Nix state directory (default `prefix/var/nix`).

- **NIX_CONF_DIR**

Overrides the location of the system Nix configuration directory (default `prefix/etc/nix`).

- **NIX_CONFIG**

Applies settings from Nix configuration from the environment. The content is treated

as if it was read from a Nix configuration file. Settings are separated by the newline character.

- [NIX_USER_CONF_FILES](#)

Overrides the location of the Nix user configuration files to load from.

The default are the locations according to the [XDG Base Directory Specification](#). See the [XDG Base Directories](#) sub-section for details.

The variable is treated as a list separated by the `:` token.

- [TMPDIR](#)

Use the specified directory to store temporary files. In particular, this includes temporary build directories; these can take up substantial amounts of disk space. The default is `/tmp`.

- [NIX_REMOTE](#)

This variable should be set to `daemon` if you want to use the Nix daemon to execute Nix operations. This is necessary in [multi-user Nix installations](#). If the Nix daemon's Unix socket is at some non-standard path, this variable should be set to `unix://path/to/socket`. Otherwise, it should be left unset.

- [NIX_SHOW_STATS](#)

If set to `1`, Nix will print some evaluation statistics, such as the number of values allocated.

- [NIX_COUNT_CALLS](#)

If set to `1`, Nix will print how often functions were called during Nix expression evaluation. This is useful for profiling your Nix expressions.

- [GC_INITIAL_HEAP_SIZE](#)

If Nix has been configured to use the Boehm garbage collector, this variable sets the initial size of the heap in bytes. It defaults to 384 MiB. Setting it to a low value reduces memory consumption, but will increase runtime due to the overhead of garbage collection.

XDG Base Directories

Nix follows the [XDG Base Directory Specification](#).

For backwards compatibility, Nix commands will follow the standard only when `use-xdg-base-directories` is enabled. [New Nix commands](#) (experimental) conform to the standard by default.

The following environment variables are used to determine locations of various state and configuration files:

- `XDG_CONFIG_HOME` (default `~/.config`)
- `XDG_STATE_HOME` (default `~/.local/state`)
- `XDG_CACHE_HOME` (default `~/.cache`)

Examples

Instantiate `store derivations`s from a Nix expression, and build them using `nix-store`:

```
$ nix-instantiate test.nix (instantiate)
/nix/store/cigxbmvyy6dzix98dxxh9b6shg7ar5bvs-perl-BerkeleyDB-0.26.drv

$ nix-store --realise $(nix-instantiate test.nix) (build)
...
/nix/store/qhqk4n8ci095g3sdp93x7rgwyh9rdvgk-perl-BerkeleyDB-0.26 (output path)

$ ls -l /nix/store/qhqk4n8ci095g3sdp93x7rgwyh9rdvgk-perl-BerkeleyDB-0.26
dr-xr-xr-x    2 eelco    users        4096 1970-01-01 01:00 lib
...
```

You can also give a Nix expression on the command line:

```
$ nix-instantiate --expr 'with import <nixpkgs> { }; hello'
/nix/store/j8s4zyv75a724q38cb0r87rlczaig4y-hello-2.8.drv
```

This is equivalent to:

```
$ nix-instantiate '<nixpkgs>' --attr hello
```

Parsing and evaluating Nix expressions:

```
$ nix-instantiate --parse --expr '1 + 2'
1 + 2

$ nix-instantiate --eval --expr '1 + 2'
3
```

```
$ nix-instantiate --eval --xml --expr '1 + 2'  
<?xml version='1.0' encoding='utf-8'?>  
<expr>  
  <int value="3" />  
</expr>
```

The difference between non-strict and strict evaluation:

```
$ nix-instantiate --eval --xml --expr 'rec { x = "foo"; y = x; }'  
...  
  <attr name="x">  
    <string value="foo" />  
  </attr>  
  <attr name="y">  
    <unevaluated />  
  </attr>  
...  
...
```

Note that `y` is left unevaluated (the XML representation doesn't attempt to show non-normal forms).

```
$ nix-instantiate --eval --xml --strict --expr 'rec { x = "foo"; y = x; }'  
...  
  <attr name="x">  
    <string value="foo" />  
  </attr>  
  <attr name="y">  
    <string value="foo" />  
  </attr>  
...  
...
```

Name

`nix-prefetch-url` - copy a file from a URL into the store and print its hash

Synopsis

```
nix-prefetch-url url [hash] [ --type hashAlgo] [ --print-path] [ --unpack] [ --name name]
```

Description

The command `nix-prefetch-url` downloads the file referenced by the URL `url`, prints its cryptographic hash, and copies it into the Nix store. The file name in the store is `hash-baseName`, where `baseName` is everything following the final slash in `url`.

This command is just a convenience for Nix expression writers. Often a Nix expression fetches some source distribution from the network using the `fetchurl` expression contained in Nixpkgs. However, `fetchurl` requires a cryptographic hash. If you don't know the hash, you would have to download the file first, and then `fetchurl` would download it again when you build your Nix expression. Since `fetchurl` uses the same name for the downloaded file as `nix-prefetch-url`, the redundant download can be avoided.

If `hash` is specified, then a download is not performed if the Nix store already contains a file with the same hash and base name. Otherwise, the file is downloaded, and an error is signaled if the actual hash of the file does not match the specified hash.

This command prints the hash on standard output. The hash is printed using base-32 unless `--type md5` is specified, in which case it's printed using base-16. Additionally, if the option `--print-path` is used, the path of the downloaded file in the Nix store is also printed.

Options

- `--type hashAlgo`

Use the specified cryptographic hash algorithm, which can be one of `md5`, `sha1`, `sha256`, and `sha512`. The default is `sha256`.

- `--print-path`

Print the store path of the downloaded file on standard output.

- **--unpack**

Unpack the archive (which must be a tarball or zip file) and add the result to the Nix store. The resulting hash can be used with functions such as Nixpkgs's `fetchzip` or `fetchFromGitHub`.

- **--executable**

Set the executable bit on the downloaded file.

- **--name *name***

Override the name of the file in the Nix store. By default, this is `hash-basename`, where `basename` is the last component of `url`. Overriding the name is necessary when `basename` contains characters that are not allowed in Nix store paths.

Examples

```
$ nix-prefetch-url ftp://ftp.gnu.org/pub/gnu/hello/hello-2.10.tar.gz  
0ssi1wpaf7plaswqqjwigppsg5fyh99vdlb9kzl7c9lnc89ndq1i
```

```
$ nix-prefetch-url --print-path mirror://gnu/hello/hello-2.10.tar.gz  
0ssi1wpaf7plaswqqjwigppsg5fyh99vdlb9kzl7c9lnc89ndq1i  
/nix/store/3x7dwzq014bbblazs7kq20p9hyzz0qh8g-hello-2.10.tar.gz
```

```
$ nix-prefetch-url --unpack --print-path https://github.com/NixOS/patchelf  
/archive/0.8.tar.gz  
079agjlv0hrv7fxnx9ngipx14gyncbkllxr9cccnh3a50fxcmmy7  
/nix/store/19zrmhm3m40xxaw81c8cqmq6aljgrnwj2-0.8.tar.gz
```

Experimental Commands

This section lists [experimental commands](#).

Warning

These commands may be removed in the future, or their syntax may change in incompatible ways.

Warning

This program is **experimental** and its interface is subject to change.

Name

nix - a tool for reproducible and declarative configuration management

Synopsis

nix [*option...*] *subcommand*

where *subcommand* is one of the following:

Help commands:

- **nix help** - show help about **nix** or a particular subcommand
- **nix help-stores** - show help about store types and their settings

Main commands:

- **nix build** - build a derivation or fetch a store path
- **nix develop** - run a bash shell that provides the build environment of a derivation
- **nix flake** - manage Nix flakes
- **nix profile** - manage Nix profiles
- **nix run** - run a Nix application
- **nix search** - search for packages
- **nix shell** - run a shell in which the specified packages are available

Main commands:

- **nix repl** - start an interactive environment for evaluating Nix expressions

Infrequently used commands:

- **nix bundle** - bundle an application so that it works outside of the Nix store
- **nix copy** - copy paths between Nix stores
- **nix edit** - open the Nix expression of a Nix package in \$EDITOR
- **nix eval** - evaluate a Nix expression

- `nix fmt` - reformat your code in the standard style
- `nix log` - show the build log of the specified packages or paths, if available
- `nix path-info` - query information about store paths
- `nix registry` - manage the flake registry
- `nix why-depends` - show why a package has another package in its closure

Utility/scripting commands:

- `nix daemon` - daemon to perform store operations on behalf of non-root clients
- `nix derivation` - Work with derivations, Nix's notion of a build plan.
- `nix hash` - compute and convert cryptographic hashes
- `nix key` - generate and convert Nix signing keys
- `nix nar` - create or inspect NAR files
- `nix print-dev-env` - print shell code that can be sourced by bash to reproduce the build environment of a derivation
- `nix realisation` - manipulate a Nix realisation
- `nix show-config` - show the Nix configuration or the value of a specific setting
- `nix store` - manipulate a Nix store

Commands for upgrading or troubleshooting your Nix installation:

- `nix doctor` - check your system for potential problems and print a PASS or FAIL for each check
- `nix upgrade-nix` - upgrade Nix to the stable version declared in Nixpkgs

Examples

- Create a new flake:

```
# nix flake new hello
# cd hello
```

- Build the flake in the current directory:

```
# nix build
# ./result/bin/hello
Hello, world!
```

- Run the flake in the current directory:

```
# nix run  
Hello, world!
```

- Start a development shell for hacking on this flake:

```
# nix develop  
# unpackPhase  
# cd hello-*  
# configurePhase  
# buildPhase  
# ./hello  
Hello, world!  
# installPhase  
# ../outputs/out/bin/hello  
Hello, world!
```

Description

Nix is a tool for building software, configurations and other artifacts in a reproducible and declarative way. For more information, see the [Nix homepage](#) or the [Nix manual](#).

Installables

Warning

Installables are part of the unstable [nix-command experimental feature](#), and subject to change without notice.

Many `nix` subcommands operate on one or more *installables*. These are command line arguments that represent something that can be realised in the Nix store.

The following types of installable are supported by most commands:

- [Flake output attribute](#) (experimental)
- [Store path](#)
- [Nix file](#), optionally qualified by an attribute path
- [Nix expression](#), optionally qualified by an attribute path

For most commands, if no installable is specified, `.` is assumed. That is, Nix will operate on the default flake output attribute of the flake in the current directory.

Flake output attribute

Warning

Flake output attribute installables depend on both the `flakes` and `nix-command` experimental features, and subject to change without notice.

Example: `nixpkgs#hello`

These have the form `flakeref[# attrpath]`, where `flakeref` is a [flake reference](#) and `attrpath` is an optional attribute path. For more information on flakes, see the [nix flake manual page](#). Flake references are most commonly a flake identifier in the flake registry (e.g. `nixpkgs`), or a raw path (e.g. `/path/to/my-flake` or `.` or `../foo`), or a full URL (e.g. `github:nixos/nixpkgs` or `path:..`)

When the flake reference is a raw path (a path without any URL scheme), it is interpreted as a `path:` or `git+file:` url in the following way:

- If the path is within a Git repository, then the url will be of the form
`git+file://[GIT_REPO_ROOT]?dir=[RELATIVE_FLAKE_DIR_PATH]` where
`GIT_REPO_ROOT` is the path to the root of the git repository, and
`RELATIVE_FLAKE_DIR_PATH` is the path (relative to the directory root) of the closest parent of the given path that contains a `flake.nix` within the git repository. If no such directory exists, then Nix will error-out.

Note that the search will only include files indexed by git. In particular, files which are matched by `.gitignore` or have never been `git add`-ed will not be available in the flake. If this is undesirable, specify `path:<directory>` explicitly;

For example, if `/foo/bar` is a git repository with the following structure:

```
.  
└── baz  
    ├── blah  
    │   └── file.txt  
    └── flake.nix
```

Then `/foo/bar/baz/blah` will resolve to `git+file:///foo/bar?dir=baz`

- If the supplied path is not a git repository, then the url will have the form `path:FLAKE_DIR_PATH` where `FLAKE_DIR_PATH` is the closest parent of the supplied path that contains a `flake.nix` file (within the same file-system). If no such directory exists, then Nix will error-out.

For example, if `/foo/bar/flake.nix` exists, then `/foo/bar/baz/` will resolve to `path:/foo/bar`

If `attrpath` is omitted, Nix tries some default values; for most subcommands, the default is `packages.system.default` (e.g. `packages.x86_64-linux.default`), but some subcommands have other defaults. If `attrpath` is specified, `attrpath` is interpreted as relative to one or more prefixes; for most subcommands, these are `packages.system`, `legacyPackages.*system*` and the empty prefix. Thus, on `x86_64-linux nix build nixpkgs#hello` will try to build the attributes `packages.x86_64-linux.hello`, `legacyPackages.x86_64-linux.hello` and `hello`.

Store path

Example: `/nix/store/v5sv61sszx301i0x6xysaqzla09nksnd-hello-2.10`

These are paths inside the Nix store, or symlinks that resolve to a path in the Nix store.

A [store derivation](#) is also addressed by store path.

Example: `/nix/store/p7gp6lxdg32h4ka1q398wd9r2zkbbz2v-hello-2.10.drv`

If you want to refer to an output path of that store derivation, add the output name preceded by a caret (`^`).

Example: `/nix/store/p7gp6lxdg32h4ka1q398wd9r2zkbbz2v-hello-2.10.drv^out`

All outputs can be referred to at once with the special syntax `^*`.

Example: `/nix/store/p7gp6lxdg32h4ka1q398wd9r2zkbbz2v-hello-2.10.drv^*`

Nix file

Example: `--file /path/to/nixpkgs hello`

When the option `-f` / `--file` `path [attrpath...]` is given, installables are interpreted as the

value of the expression in the Nix file at *path*. If attribute paths are provided, commands will operate on the corresponding values accessible at these paths. The Nix expression in that file, or any selected attribute, must evaluate to a derivation.

Nix expression

Example: `--expr 'import <nixpkgs> {}' hello`

When the option `--expr expression [attrpath...]` is given, installables are interpreted as the value of the of the Nix expression. If attribute paths are provided, commands will operate on the corresponding values accessible at these paths. The Nix expression, or any selected attribute, must evaluate to a derivation.

You may need to specify `--impure` if the expression references impure inputs (such as `<nixpkgs>`).

Derivation output selection

Derivations can have multiple outputs, each corresponding to a different store path. For instance, a package can have a `bin` output that contains programs, and a `dev` output that provides development artifacts like C/C++ header files. The outputs on which `nix` commands operate are determined as follows:

- You can explicitly specify the desired outputs using the syntax *installable* `^ output1 , ... , outputN`. For example, you can obtain the `dev` and `static` outputs of the `glibc` package:

```
# nix build 'nixpkgs#glibc^dev,static'  
# ls ./result-dev/include/ ./result-static/lib/  
...
```

and likewise, using a store path to a "drv" file to specify the derivation:

```
# nix build '/nix/store/gzaflydcr6sb3567hap9q6srzx8ggdgg-glibc-  
2.33-78.drv^dev,static'  
...
```

- You can also specify that *all* outputs should be used using the syntax *installable* `^*`. For example, the following shows the size of all outputs of the `glibc` package in the binary cache:

```
# nix path-info --closure-size --eval-store auto --store
https://cache.nixos.org 'nixpkgs#glibc^*'
/nix/store/g02b1lpbddhymmcjb923kf0l7s9nww58-glibc-2.33-123
33208200
/nix/store/851dp95qqiisjifi639r0zzg5l465ny4-glibc-2.33-123-bin
36142896
/nix/store/kdgs3q6r7xdff1p7a9hnjr43xw2404z7-glibc-2.33-123-debug
155787312
/nix/store/n4xa8h6pbmqmwnq0mmsz08l38abb06zc-glibc-2.33-123-static
42488328
/nix/store/q6580lr01jpcsq4r5arlh4ki2c1m9rv-glibc-2.33-123-dev
44200560
```

and likewise, using a store path to a "drv" file to specify the derivation:

```
# nix path-info --closure-size '/nix/store
/gzaflydcr6sb3567hap9q6srzx8ggdgg-glibc-2.33-78.drv^*'
...
...
```

- If you didn't specify the desired outputs, but the derivation has an attribute `meta.outputsToInstall`, Nix will use those outputs. For example, since the package `nixpkgs#libxml2` has this attribute:

```
# nix eval 'nixpkgs#libxml2.meta.outputsToInstall'
[ "bin" "man" ]
```

a command like `nix shell nixpkgs#libxml2` will provide only those two outputs by default.

Note that a [store derivation](#) (given by its `.drv` file store path) doesn't have any attributes like `meta`, and thus this case doesn't apply to it.

- Otherwise, Nix will use all outputs of the derivation.

Nix stores

Most `nix` subcommands operate on a *Nix store*. These are documented in [nix help-stores](#).

Options

Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-`

`code-during-evaluation` setting.

- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.

- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.

- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.
- `--max-silent-time` *value* Set the `max-silent-time` setting.
- `--max-substitution-jobs` *value* Set the `max-substitution-jobs` setting.
- `--min-free` *value* Set the `min-free` setting.
- `--min-free-check-interval` *value* Set the `min-free-check-interval` setting.
- `--nar-buffer-size` *value* Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl` *value* Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl` *value* Set the `narinfo-cache-positive-ttl` setting.

- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.

- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.

- `--post-build-hook` *value* Set the `post-build-hook` setting.
- `--pre-build-hook` *value* Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.

- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

Warning

This program is **experimental** and its interface is subject to change.

Name

`nix build` - build a derivation or fetch a store path

Synopsis

`nix build [option...] installables...`

Examples

- Build the default package from the flake in the current directory:

```
# nix build
```

- Build and run GNU Hello from the `nixpkgs` flake:

```
# nix build nixpkgs#hello
# ./result/bin/hello
Hello, world!
```

- Build GNU Hello and Cowsay, leaving two result symlinks:

```
# nix build nixpkgs#hello nixpkgs#cowsay
# ls -l result*
lrwxrwxrwx 1 ... result -> /nix/store/v5sv61sszx301i0x6xysaqzla09nksnd-
hello-2.10
lrwxrwxrwx 1 ... result-1 -> /nix/store/rkfrm0z6x6jmi7d3gsmma4j53h15mg33-
cowsay-3.03+dfsg2
```

- Build GNU Hello and print the resulting store path.

```
# nix build nixpkgs#hello --print-out-paths
/nix/store/v5sv61sszx301i0x6xysaqzla09nksnd-hello-2.10
```

- Build a specific output:

```
# nix build nixpkgs#glibc.dev
# ls -ld ./result-dev
lrwxrwxrwx 1 ... ./result-dev -> /nix/store/dkm3gwl0xrx0wrw6zi5x3px3lpgjhlw4-
glibc-2.32-dev
```

- Build attribute `build.x86_64-linux` from (non-flake) Nix expression `release.nix`:

```
# nix build --file release.nix build.x86_64-linux
```

- Build a NixOS system configuration from a flake, and make a profile point to the result:

```
# nix build --profile /nix/var/nix/profiles/system \
~/my-
configurations#nixosConfigurations.machine.config.system.build.toplevel
```

(This is essentially what `nixos-rebuild` does.)

- Build an expression specified on the command line:

```
# nix build --impure --expr \
'with import <nixpkgs> {};
runCommand "foo" {
  buildInputs = [ hello ];
}
"hello > $out"
# cat ./result
Hello, world!
```

Note that `--impure` is needed because we're using `<nixpkgs>`, which relies on the `$NIX_PATH` environment variable.

- Fetch a store path from the configured substituters, if it doesn't already exist:

```
# nix build /nix/store/rkfrm0z6x6jmi7d3gsmma4j53h15mg33-cowsay-3.03+dfsg2
```

Description

`nix build` builds the specified *installables*. **Installables** that resolve to derivations are built (or substituted if possible). Store path installables are substituted.

Unless `--no-link` is specified, after a successful build, it creates symlinks to the store paths of the installables. These symlinks have the prefix `./result` by default; this can be overridden using the `--out-link` option. Each symlink has a suffix `-<N>-<outname>`, where N is the index of the installable (with the left-most installable having index 0), and *outname* is the symbolic derivation output name (e.g. `bin`, `dev` or `lib`). `-<N>` is omitted if $N = 0$, and `-<outname>` is omitted if *outname* = `out` (denoting the default output).

Options

- `--dry-run` Show what this command would do without doing it.
- `--json` Produce output in JSON format, suitable for consumption by another program.
- `--no-link` Do not create symlinks to the build results.
- `--out-link / -o path` Use *path* as prefix for the symlinks to the build results. It defaults to `result`.
- `--print-out-paths` Print the resulting output paths
- `--profile path` The profile to operate on.
- `--rebuild` Rebuild an already built package and compare the result to the existing store paths.
- `--stdin` Read installables from the standard input. No default installable applied.

Common evaluation options:

- `--arg name expr` Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr name string` Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store store-url` The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.

- `--impure` Allow access to mutable paths and repositories.
- `--include / -I path` Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev  
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch  
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05  
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

Options that change the interpretation of `installables`:

- `--expr` *expr* Interpret `installables` as attribute paths relative to the Nix expression *expr*.
- `--file` / `-f` *file* Interpret `installables` as attribute paths relative to the Nix expression stored in *file*. If *file* is the character -, then a Nix expression will be read from standard input. Implies `--impure`.

Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.

- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.

- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.

- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.
- `--max-silent-time` *value* Set the `max-silent-time` setting.
- `--max-substitution-jobs` *value* Set the `max-substitution-jobs` setting.
- `--min-free` *value* Set the `min-free` setting.
- `--min-free-check-interval` *value* Set the `min-free-check-interval` setting.

- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.

- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.

- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.
- `--start-id value` Set the `start-id` setting.

- `--store value` Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

Warning

This program is **experimental** and its interface is subject to change.

Name

`nix bundle` - bundle an application so that it works outside of the Nix store

Synopsis

`nix bundle [option...] installable`

Examples

- Bundle Hello:

```
# nix bundle nixpkgs#hello
# ./hello
Hello, world!
```

- Bundle a specific version of Nix:

```
# nix bundle github:NixOS/nix/e3ddff27e5fc37a209cf843c6f7f6a9460a8ec
# ./nix --version
nix (Nix) 2.4pre20201215_e3ddff2
```

- Bundle a Hello using a specific bundler:

```
# nix bundle --bundler github:NixOS/bundlers#toDockerImage nixpkgs#hello
# docker load < hello-2.10.tar.gz
# docker run hello-2.10:latest hello
Hello, world!
```

Description

`nix bundle`, by default, packs the closure of the *installable* into a single self-extracting executable. See the [bundlers homepage](#) for more details.

Note

This command only works on Linux.

Flake output attributes

If no flake output attribute is given, `nix bundle` tries the following flake output attributes:

- `bundlers.<system>.default`

If an attribute *name* is given, `nix bundle` tries the following flake output attributes:

- `bundlers.<system>.<name>`

Bundlers

A bundler is specified by a flake output attribute named `bundlers.<system>.<name>`. It looks like this:

```
bundlers.x86_64-linux = rec {
  identity = drv: drv;

  blender_2_79 = drv: self.packages.x86_64-linux.blender_2_79;

  default = identity;
};
```

A bundler must be a function that accepts an arbitrary value (typically a derivation or app definition) and returns a derivation.

Options

- `--bundler flake-url` Use a custom bundler instead of the default (`github:NixOS/bundlers`).
- `--out-link / -o path` Override the name of the symlink to the build result. It defaults to the base name of the app.

Common evaluation options:

- `--arg name expr` Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr name string` Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store store-url` The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include / -I path` Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev  
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch  
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location.

The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05  
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.

- **--output-lock-file** *flake-lock-path* Write the given lock file instead of `flake.lock` within the top-level flake.
- **--override-input** *input-path flake-url* Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies **--no-write-lock-file**.
- **--recreate-lock-file** Recreate the flake's lock file from scratch.
- **--reference-lock-file** *flake-lock-path* Read the given lock file instead of `flake.lock` within the top-level flake.
- **--update-input** *input-path* Update a specific flake input (ignoring its previous entry in the lock file).

Logging-related options:

- **--debug** Set the logging verbosity level to 'debug'.
- **--log-format** *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- **--print-build-logs** / **-L** Print full build logs on standard error.
- **--quiet** Decrease the logging verbosity level.
- **--verbose** / **-v** Increase the logging verbosity level.

Miscellaneous global options:

- **--help** Show usage information.
- **--offline** Disable substituters and consider all previously downloaded files up-to-date.
- **--option** *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- **--refresh** Consider all previously downloaded files out-of-date.
- **--repair** During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- **--version** Show version information.

Options that change the interpretation of **installables**:

- **--expr** *expr* Interpret *installables* as attribute paths relative to the Nix expression *expr*.
- **--file** / **-f** *file* Interpret *installables* as attribute paths relative to the Nix expression stored in *file*. If *file* is the character -, then a Nix expression will be read from standard input. Implies **--impure**.

Options to override configuration settings:

- **--accept-flake-config** Enable the `accept-flake-config` setting.
- **--access-tokens** *value* Set the `access-tokens` setting.
- **--allow-dirty** Enable the `allow-dirty` setting.
- **--allow-import-from-derivation** Enable the `allow-import-from-derivation` setting.
- **--allow-new-privileges** Enable the `allow-new-privileges` setting.
- **--allow-symlinked-store** Enable the `allow-symlinked-store` setting.
- **--allow-unsafe-native-code-during-evaluation** Enable the `allow-unsafe-native-code-during-evaluation` setting.
- **--allowed-impure-host-deps** *value* Set the `allowed-impure-host-deps` setting.
- **--allowed-uris** *value* Set the `allowed-uris` setting.
- **--allowed-users** *value* Set the `allowed-users` setting.
- **--auto-allocate-uids** Enable the `auto-allocate-uids` setting.
- **--auto-optimise-store** Enable the `auto-optimise-store` setting.
- **--bash-prompt** *value* Set the `bash-prompt` setting.
- **--bash-prompt-prefix** *value* Set the `bash-prompt-prefix` setting.
- **--bash-prompt-suffix** *value* Set the `bash-prompt-suffix` setting.
- **--build-hook** *value* Set the `build-hook` setting.
- **--build-poll-interval** *value* Set the `build-poll-interval` setting.
- **--build-users-group** *value* Set the `build-users-group` setting.

- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.

- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.

- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.

- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.

- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.

- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.

- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

Warning

This program is **experimental** and its interface is subject to change.

Name

`nix copy` - copy paths between Nix stores

Synopsis

`nix copy [option...] installables...`

Examples

- Copy Firefox from the local store to a binary cache in `/tmp/cache`:

```
# nix copy --to file:///tmp/cache $(type -p firefox)
```

Note the `file://` - without this, the destination is a chroot store, not a binary cache.

- Copy the entire current NixOS system closure to another machine via SSH:

```
# nix copy --substitute-on-destination --to ssh://server /run/current-system
```

The `-s` flag causes the remote machine to try to substitute missing store paths, which may be faster if the link between the local and remote machines is slower than the link between the remote machine and its substituters (e.g. <https://cache.nixos.org>).

- Copy a closure from another machine via SSH:

```
# nix copy --from ssh://server /nix/store/a6cnl93nk1wxnq84brbbwr6hxw9gp2w9-blender-2.79-rc2
```

- Copy Hello to a binary cache in an Amazon S3 bucket:

```
# nix copy --to s3://my-bucket?region=eu-west-1 nixpkgs#hello
```

or to an S3-compatible storage system:

```
# nix copy --to s3://my-bucket?region=eu-west-1&endpoint=example.com  
nixpkgs#hello
```

Note that this only works if Nix is built with AWS support.

- Copy a closure from `/nix/store` to the chroot store `/tmp/nix/nix/store`:

```
# nix copy --to /tmp/nix nixpkgs#hello --no-check-sigs
```

Description

`nix copy` copies store path closures between two Nix stores. The source store is specified using `--from` and the destination using `--to`. If one of these is omitted, it defaults to the local store.

Options

- `--from store-uri` URL of the source Nix store.
- `--no-check-sigs` Do not require that paths are signed by trusted keys.
- `--stdin` Read installables from the standard input. No default installable applied.
- `--substitute-on-destination / -s` Whether to try substitutes on the destination store (only supported by SSH stores).
- `--to store-uri` URL of the destination Nix store.

Common evaluation options:

- `--arg name expr` Pass the value *expr* as the argument *name* to Nix functions.
- `--argsstr name string` Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.

- `--eval-store store-url` The URL of the Nix store to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include / -I path` Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev  
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch  
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05  
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.

- `--update-input` *input-path* Update a specific flake input (ignoring its previous entry in the lock file).

Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

Options that change the interpretation of `installables`:

- `--all` Apply the operation to every store path.
- `--derivation` Operate on the `store derivation` rather than its outputs.
- `--expr` *expr* Interpret `installables` as attribute paths relative to the Nix expression *expr*.
- `--file` / `-f` *file* Interpret `installables` as attribute paths relative to the Nix expression stored in *file*. If *file* is the character -, then a Nix expression will be read from standard input. Implies `--impure`.
- `--no-recursive` Apply operation to specified paths only.

Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.

- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors value` Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls value` Append to the `ignored-acls` setting.
- `--extra-nix-path value` Append to the `nix-path` setting.
- `--extra-platforms value` Set the `extra-platforms` setting.
- `--extra-plugin-files value` Append to the `plugin-files` setting.
- `--extra-sandbox-paths value` Append to the `sandbox-paths` setting.
- `--extra-secret-key-files value` Append to the `secret-key-files` setting.
- `--extra-substituters value` Append to the `substituters` setting.
- `--extra-system-features value` Append to the `system-features` setting.

- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.

- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.

- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.

- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.

- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.
- `--start-id value` Set the `start-id` setting.
- `--store value` Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.

- `--warn-dirty` Enable the `warn-dirty` setting.

Warning

This program is **experimental** and its interface is subject to change.

Name

`nix daemon` - daemon to perform store operations on behalf of non-root clients

Synopsis

`nix daemon` [*option...*]

Examples

- Run the daemon:

```
# nix daemon
```

- Run the daemon and listen on standard I/O instead of binding to a UNIX socket:

```
# nix daemon --stdio
```

- Run the daemon and force all connections to be trusted:

```
# nix daemon --force-trusted
```

- Run the daemon and force all connections to be untrusted:

```
# nix daemon --force-untrusted
```

- Run the daemon, listen on standard I/O, and force all connections to use Nix's default trust:

```
# nix daemon --stdio --default-trust
```

Description

This command runs the Nix daemon, which is a required component in multi-user Nix installations. It runs build tasks and other operations on the Nix store on behalf of non-root users. Usually you don't run the daemon directly; instead it's managed by a service management framework such as `systemd` on Linux, or `launchctl` on Darwin.

Note that this daemon does not fork into the background.

Options

- `--default-trust` Use Nix's default trust.
- `--force-trusted` Force the daemon to trust connecting clients.
- `--force-untrusted` Force the daemon to not trust connecting clients. The connection will be processed by the receiving daemon before forwarding commands.
- `--stdio` Attach to standard I/O, instead of trying to bind to a UNIX socket.

Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).

- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.

- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors value` Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls value` Append to the `ignored-acls` setting.
- `--extra-nix-path value` Append to the `nix-path` setting.
- `--extra-platforms value` Set the `extra-platforms` setting.
- `--extra-plugin-files value` Append to the `plugin-files` setting.
- `--extra-sandbox-paths value` Append to the `sandbox-paths` setting.
- `--extra-secret-key-files value` Append to the `secret-key-files` setting.

- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.

- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.

- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.

- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.

- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.

- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

Warning

This program is **experimental** and its interface is subject to change.

Name

nix derivation - Work with derivations, Nix's notion of a build plan.

Synopsis

nix derivation [*option...*] *subcommand*

where *subcommand* is one of the following:

- **nix derivation add** - Add a store derivation
- **nix derivation show** - show the contents of a store derivation

Warning

This program is **experimental** and its interface is subject to change.

Name

`nix derivation add` - Add a store derivation

Synopsis

`nix derivation add [option...]`

Description

This command reads from standard input a JSON representation of a [store derivation](#) to which an [installable](#) evaluates.

Store derivations are used internally by Nix. They are store paths with extension `.drv` that represent the build-time dependency graph to which a Nix expression evaluates.

The JSON format is documented under the [derivation show](#) command.

Options

- `--dry-run` Show what this command would do without doing it.

Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.

- `--verbose` / `-v` Increase the logging verbosity level.

Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.

- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.
- `--build-hook value` Set the `build-hook` setting.
- `--build-poll-interval value` Set the `build-poll-interval` setting.
- `--build-users-group value` Set the `build-users-group` setting.
- `--builders value` Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.

- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.

- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.

- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.

- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-`

groups setting.

- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.

- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

Warning

This program is **experimental** and its interface is subject to change.

Name

`nix derivation show` - show the contents of a store derivation

Synopsis

`nix derivation show [option...] installables...`

Examples

- Show the **store derivation** that results from evaluating the Hello package:

```
# nix derivation show nixpkgs#hello
{
  "/nix/store/s6rn4jz1sin56rf4qj5b5v8jxjm32hlk-hello-2.10.drv": {
    ...
  }
}
```

- Show the full derivation graph (if available) that produced your NixOS system:

```
# nix derivation show -r /run/current-system
```

- Print all files fetched using **fetchurl** by Firefox's dependency graph:

```
# nix derivation show -r nixpkgs#firefox |
  | jq -r '.[] | select(.outputs.out.hash and .env.urls) | .env.urls' \
  | uniq | sort
```

Note that `.outputs.out.hash` selects *fixed-output derivations* (derivations that produce output with a specified content hash), while `.env.urls` selects derivations with a `urls`

attribute.

Description

This command prints on standard output a JSON representation of the [store derivations](#) to which [installables](#) evaluate.

Store derivations are used internally by Nix. They are store paths with extension `.drv` that represent the build-time dependency graph to which a Nix expression evaluates.

By default, this command only shows top-level derivations, but with `--recursive`, it also shows their dependencies.

The JSON output is a JSON object whose keys are the store paths of the derivations, and whose values are a JSON object with the following fields:

- `name` : The name of the derivation. This is used when calculating the store paths of the derivation's outputs.
- `outputs` : Information about the output paths of the derivation. This is a JSON object with one member per output, where the key is the output name and the value is a JSON object with these fields:
 - `path` : The output path.
 - `hashAlgo` : For fixed-output derivations, the hashing algorithm (e.g. `sha256`), optionally prefixed by `r:` if `hash` denotes a NAR hash rather than a flat file hash.
 - `hash` : For fixed-output derivations, the expected content hash in base-16.

Example:

```
"outputs": {
    "out": {
        "path": "/nix/store/2543j7c6jn75blc3drf4g5vhb1rhdq29-source",
        "hashAlgo": "r:sha256",
        "hash":
"6fc80dcc62179dbc12fc0b5881275898f93444833d21b89dfe5f7fbcb1d0d62"
    }
}
```

- `inputSrcs` : A list of store paths on which this derivation depends.

- `inputDrv`: A JSON object specifying the derivations on which this derivation depends, and what outputs of those derivations. For example,

```
"inputDrv": {  
    "/nix/store/6lkh5yi7nlb7l6dr8fljlli5zfd9hq58-curl-7.73.0.drv": ["dev"],  
    "/nix/store/fn3kgnfzl5dzym26j8g907gq3kbm8bfh-unzip-6.0.drv": ["out"]  
}
```

specifies that this derivation depends on the `dev` output of `curl`, and the `out` output of `unzip`.

- `system`: The system type on which this derivation is to be built (e.g. `x86_64-linux`).
- `builder`: The absolute path of the program to be executed to run the build. Typically this is the `bash` shell (e.g. `/nix/store/r3j288vpmczbl500w6zz89gyfa4nr0b1-bash-4.4-p23/bin/bash`).
- `args`: The command-line arguments passed to the `builder`.
- `env`: The environment passed to the `builder`.

Options

- `--recursive` / `-r` Include the dependencies of the specified derivations.
- `--stdin` Read installables from the standard input. No default installable applied.

Common evaluation options:

- `--arg` *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr` *name string* Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store` *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include` / `-I` *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the

location of Nix expressions using [paths](#) enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev  
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch  
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for nix-channel](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05  
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.

- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

Options that change the interpretation of `installables`:

- `--expr expr` Interpret *installables* as attribute paths relative to the Nix expression *expr*.
- `--file` / `-f file` Interpret *installables* as attribute paths relative to the Nix expression stored in *file*. If *file* is the character -, then a Nix expression will be read from standard input. Implies `--impure`.

Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.

- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps value` Set the `allowed-impure-host-deps` setting.
- `--allowed-uris value` Set the `allowed-uris` setting.
- `--allowed-users value` Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt value` Set the `bash-prompt` setting.
- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.
- `--build-hook value` Set the `build-hook` setting.
- `--build-poll-interval value` Set the `build-poll-interval` setting.
- `--build-users-group value` Set the `build-users-group` setting.
- `--builders value` Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.

- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.

- `--gc-reserved-space value` Set the `gc-reserved-space` setting.
- `--hashed-mirrors value` Set the `hashed-mirrors` setting.
- `--http-connections value` Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count value` Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls value` Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.

- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.

- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.

- `--plugin-files` *value* Set the `plugin-files` setting.
- `--post-build-hook` *value* Set the `post-build-hook` setting.
- `--pre-build-hook` *value* Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.

- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

Warning

This program is **experimental** and its interface is subject to change.

Name

`nix develop` - run a bash shell that provides the build environment of a derivation

Synopsis

`nix develop [option...] installable`

Examples

- Start a shell with the build environment of the default package of the flake in the current directory:

```
# nix develop
```

Typical commands to run inside this shell are:

```
# configurePhase  
# buildPhase  
# installPhase
```

Alternatively, you can run whatever build tools your project uses directly, e.g. for a typical Unix project:

```
# ./configure --prefix=$out  
# make  
# make install
```

- Run a particular build phase directly:

```
# nix develop --unpack  
# nix develop --configure  
# nix develop --build  
# nix develop --check  
# nix develop --install  
# nix develop --installcheck
```

- Start a shell with the build environment of GNU Hello:

```
# nix develop nixpkgs#hello
```

- Record a build environment in a profile:

```
# nix develop --profile /tmp/my-build-env nixpkgs#hello
```

- Use a build environment previously recorded in a profile:

```
# nix develop /tmp/my-build-env
```

- Replace all occurrences of the store path corresponding to `glibc.dev` with a writable directory:

```
# nix develop --redirect nixpkgs#glibc.dev ~/my-glibc/outputs/dev
```

Note that this is useful if you're running a `nix develop` shell for `nixpkgs#glibc` in `~/my-glibc` and want to compile another package against it.

- Run a series of script commands:

```
# nix develop --command bash -c "mkdir build && cmake .. && make"
```

Description

`nix develop` starts a `bash` shell that provides an interactive build environment nearly identical to what Nix would use to build `installable`. Inside this shell, environment variables and shell functions are set up so that you can interactively and incrementally build your package.

Nix determines the build environment by building a modified version of the derivation *installable* that just records the environment initialised by `stdenv` and exits. This build environment can be recorded into a profile using `--profile`.

The prompt used by the `bash` shell can be customised by setting the `bash-prompt`, `bash-prompt-prefix`, and `bash-prompt-suffix` settings in `nix.conf` or in the flake's `nixConfig` attribute.

Flake output attributes

If no flake output attribute is given, `nix develop` tries the following flake output attributes:

- `devShells.<system>.default`
- `packages.<system>.default`

If a flake output *name* is given, `nix develop` tries the following flake output attributes:

- `devShells.<system>.<name>`
- `packages.<system>.<name>`
- `legacyPackages.<system>.<name>`

Options

- `--build` Run the `build` phase.
- `--check` Run the `check` phase.
- `--command / -c command args` Instead of starting an interactive shell, start the specified command and arguments.
- `--configure` Run the `configure` phase.
- `--ignore-environment / -i` Clear the entire environment (except those specified with `--keep`).
- `--install` Run the `install` phase.
- `--installcheck` Run the `installcheck` phase.

- `--keep / -k name` Keep the environment variable *name*.
- `--phase phase-name` The stdenv phase to run (e.g. `build` or `configure`).
- `--profile path` The profile to operate on.
- `--redirect installable outputs-dir` Redirect a store path to a mutable location.
- `--unpack` Run the `unpack` phase.
- `--unset / -u name` Unset the environment variable *name*.

Common evaluation options:

- `--arg name expr` Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr name string` Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store store-url` The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include / -I path` Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev  
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch  
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05  
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.

- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file` *flake-lock-path* Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input` *input-path flake-url* Override a specific flake input (e.g. `dwarfss/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file` *flake-lock-path* Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input` *input-path* Update a specific flake input (ignoring its previous entry in the lock file).

Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.

- **--repair** During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- **--version** Show version information.

Options that change the interpretation of `installables`:

- **--expr** *expr* Interpret `installables` as attribute paths relative to the Nix expression *expr*.
- **--file** / **-f** *file* Interpret `installables` as attribute paths relative to the Nix expression stored in *file*. If *file* is the character -, then a Nix expression will be read from standard input. Implies **--impure**.

Options to override configuration settings:

- **--accept-flake-config** Enable the `accept-flake-config` setting.
- **--access-tokens** *value* Set the `access-tokens` setting.
- **--allow-dirty** Enable the `allow-dirty` setting.
- **--allow-import-from-derivation** Enable the `allow-import-from-derivation` setting.
- **--allow-new-privileges** Enable the `allow-new-privileges` setting.
- **--allow-symlinked-store** Enable the `allow-symlinked-store` setting.
- **--allow-unsafe-native-code-during-evaluation** Enable the `allow-unsafe-native-code-during-evaluation` setting.
- **--allowed-impure-host-deps** *value* Set the `allowed-impure-host-deps` setting.
- **--allowed-uris** *value* Set the `allowed-uris` setting.
- **--allowed-users** *value* Set the `allowed-users` setting.
- **--auto-allocate-uids** Enable the `auto-allocate-uids` setting.
- **--auto-optimise-store** Enable the `auto-optimise-store` setting.
- **--bash-prompt** *value* Set the `bash-prompt` setting.
- **--bash-prompt-prefix** *value* Set the `bash-prompt-prefix` setting.
- **--bash-prompt-suffix** *value* Set the `bash-prompt-suffix` setting.

- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.

- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.

- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.

- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.

- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.

- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.

- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

Warning

This program is **experimental** and its interface is subject to change.

Name

nix doctor - check your system for potential problems and print a PASS or FAIL for each check

Synopsis

nix doctor [*option...*]

Warning

This program is **experimental** and its interface is subject to change.

Name

`nix edit` - open the Nix expression of a Nix package in \$EDITOR

Synopsis

`nix edit [option...] installable`

Examples

- Open the Nix expression of the GNU Hello package:

```
# nix edit nixpkgs#hello
```

- Get the filename and line number used by `nix edit`:

```
# nix eval --raw nixpkgs#hello.meta.position
/nix/store/fvafw0gvwayzdan642wrv84pzm5bgpmy-source/pkgs/applications
/misc/hello/default.nix:15
```

Description

This command opens the Nix expression of a derivation in an editor. The filename and line number of the derivation are taken from its `meta.position` attribute. Nixpkgs' `stdenv.mkDerivation` sets this attribute to the location of the definition of the `meta.description`, `version` or `name` derivation attributes.

The editor to invoke is specified by the `EDITOR` environment variable. It defaults to `cat`. If the editor is `emacs`, `nano`, `vim` or `kak`, it is passed the line number of the derivation using

the argument `+<lineno>`.

Options

Common evaluation options:

- `--arg` *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr` *name string* Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store` *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include` / `-I` *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev  
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch  
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as

the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05  
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.

- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarfss/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

Options that change the interpretation of `installables`:

- `--expr expr` Interpret `installables` as attribute paths relative to the Nix expression `expr`.
- `--file / -f file` Interpret `installables` as attribute paths relative to the Nix expression stored in `file`. If `file` is the character `-`, then a Nix expression will be read from standard input. Implies `--impure`.

Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps value` Set the `allowed-impure-host-deps` setting.
- `--allowed-uris value` Set the `allowed-uris` setting.
- `--allowed-users value` Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt value` Set the `bash-prompt` setting.
- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.
- `--build-hook value` Set the `build-hook` setting.
- `--build-poll-interval value` Set the `build-poll-interval` setting.

- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.

- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.

- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.

- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.

- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.

- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.
- `--start-id value` Set the `start-id` setting.
- `--store value` Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.

- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

Warning

This program is **experimental** and its interface is subject to change.

Name

`nix eval` - evaluate a Nix expression

Synopsis

`nix eval` [*option...*] *installable*

Examples

- Evaluate a Nix expression given on the command line:

```
# nix eval --expr '1 + 2'
```

- Evaluate a Nix expression to JSON:

```
# nix eval --json --expr '{ x = 1; }'  
{"x":1}
```

- Evaluate a Nix expression from a file:

```
# nix eval --file ./my-nixpkgs hello.name
```

- Get the current version of the `nixpkgs` flake:

```
# nix eval --raw nixpkgs#lib.version
```

- Print the store path of the Hello package:

```
# nix eval --raw nixpkgs#hello
```

- Get a list of checks in the `nix` flake:

```
# nix eval nix#checks.x86_64-linux --apply builtins.attrNames
```

- Generate a directory with the specified contents:

```
# nix eval --write-to ./out --expr '{ foo = "bar"; subdir.bla = "123"; }'  
# cat ./out/foo  
bar  
# cat ./out/subdir/bla  
123
```

Description

This command evaluates the given Nix expression and prints the result on standard output.

Output format

`nix eval` can produce output in several formats:

- By default, the evaluation result is printed as a Nix expression.
- With `--json`, the evaluation result is printed in JSON format. Note that this fails if the result contains values that are not representable as JSON, such as functions.
- With `--raw`, the evaluation result must be a string, which is printed verbatim, without any quoting.
- With `--write-to path`, the evaluation result must be a string or a nested attribute set whose leaf values are strings. These strings are written to files named `path/attrpath`. `path` must not already exist.

Options

- `--apply expr` Apply the function `expr` to each argument.

- `--json` Produce output in JSON format, suitable for consumption by another program.
- `--raw` Print strings without quotes or escaping.
- `--read-only` Do not instantiate each evaluated derivation. This improves performance, but can cause errors when accessing store paths of derivations during evaluation.
- `--write-to path` Write a string or attrset of strings to *path*.

Common evaluation options:

- `--arg name expr` Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr name string` Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store store-url` The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include / -I path` Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev  
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch  
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-`

`branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05  
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated;

- ```
use --no-use-registries .
```
- **--no-update-lock-file** Do not allow any updates to the flake's lock file.
  - **--no-write-lock-file** Do not write the flake's newly generated lock file.
  - **--output-lock-file** *flake-lock-path* Write the given lock file instead of `flake.lock` within the top-level flake.
  - **--override-input** *input-path* *flake-url* Override a specific flake input (e.g. `dwarfes/nixpkgs`). This implies **--no-write-lock-file**.
  - **--recreate-lock-file** Recreate the flake's lock file from scratch.
  - **--reference-lock-file** *flake-lock-path* Read the given lock file instead of `flake.lock` within the top-level flake.
  - **--update-input** *input-path* Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- **--debug** Set the logging verbosity level to 'debug'.
- **--log-format** *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- **--print-build-logs** / **-L** Print full build logs on standard error.
- **--quiet** Decrease the logging verbosity level.
- **--verbose** / **-v** Increase the logging verbosity level.

## Miscellaneous global options:

- **--help** Show usage information.
- **--offline** Disable substituters and consider all previously downloaded files up-to-date.
- **--option** *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- **--refresh** Consider all previously downloaded files out-of-date.
- **--repair** During evaluation, rewrite missing or corrupted files in the Nix store. During

building, rebuild missing or corrupted store paths.

- `--version` Show version information.

## Options that change the interpretation of `installables`:

- `--expr expr` Interpret `installables` as attribute paths relative to the Nix expression `expr`.
- `--file / -f file` Interpret `installables` as attribute paths relative to the Nix expression stored in `file`. If `file` is the character `-`, then a Nix expression will be read from standard input. Implies `--impure`.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps value` Set the `allowed-impure-host-deps` setting.
- `--allowed-uris value` Set the `allowed-uris` setting.
- `--allowed-users value` Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt value` Set the `bash-prompt` setting.
- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.

- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.

- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.

- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.

- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.

- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.

- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.

- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix flake` - manage Nix flakes

## Synopsis

`nix flake [option...] subcommand`

where *subcommand* is one of the following:

- `nix flake archive` - copy a flake and all its inputs to a store
- `nix flake check` - check whether the flake evaluates and run its tests
- `nix flake clone` - clone flake repository
- `nix flake info` - show flake metadata
- `nix flake init` - create a flake in the current directory from a template
- `nix flake lock` - create missing lock file entries
- `nix flake metadata` - show flake metadata
- `nix flake new` - create a flake in the specified directory from a template
- `nix flake prefetch` - download the source tree denoted by a flake reference into the Nix store
- `nix flake show` - show the outputs provided by a flake
- `nix flake update` - update flake lock file

# Description

`nix flake` provides subcommands for creating, modifying and querying *Nix flakes*. Flakes are the unit for packaging Nix code in a reproducible and discoverable way. They can have dependencies on other flakes, making it possible to have multi-repository Nix projects.

A flake is a filesystem tree (typically fetched from a Git repository or a tarball) that contains a file named `flake.nix` in the root directory. `flake.nix` specifies some metadata about the flake such as dependencies (called *inputs*), as well as its *outputs* (the Nix values such as

packages or NixOS modules provided by the flake).

# Flake references

Flake references (*flakeref*s) are a way to specify the location of a flake. These have two different forms:

## Attribute set representation

Example:

```
{
 type = "github";
 owner = "NixOS";
 repo = "nixpkgs";
}
```

The only required attribute is `type`. The supported types are listed below.

## URL-like syntax

Example:

```
github:NixOS/nixpkgs
```

These are used on the command line as a more convenient alternative to the attribute set representation. For instance, in the command

```
nix build github:NixOS/nixpkgs#hello
```

`github:NixOS/nixpkgs` is a flake reference (while `hello` is an output attribute). They are also allowed in the `inputs` attribute of a flake, e.g.

```
inputs.nixpkgs.url = "github:NixOS/nixpkgs";
```

is equivalent to

```
inputs.nixpkgs = {
 type = "github";
 owner = "NixOS";
 repo = "nixpkgs";
};
```

## Examples

Here are some examples of flake references in their URL-like representation:

- `nixpkgs`: The `nixpkgs` entry in the flake registry.
- `nixpkgs/a3a3dda3bacf61e8a39258a0ed9c924eeca8e293`: The `nixpkgs` entry in the flake registry, with its Git revision overridden to a specific value.
- `github:NixOS/nixpkgs`: The `master` branch of the `NixOS/nixpkgs` repository on GitHub.
- `github:NixOS/nixpkgs/nixos-20.09`: The `nixos-20.09` branch of the `nixpkgs` repository.
- `github:NixOS/nixpkgs/a3a3dda3bacf61e8a39258a0ed9c924eeca8e293`: A specific revision of the `nixpkgs` repository.
- `github:edolstra/nix-warez?dir=blender`: A flake in a subdirectory of a GitHub repository.
- `git+https://github.com/NixOS/patchelf`: A Git repository.
- `git+https://github.com/NixOS/patchelf?ref=master`: A specific branch of a Git repository.
- `git+https://github.com/NixOS/patchelf?ref=master&rev=f34751b88bd07d7f44f5cd3200fb4122bf916c7e`: A specific branch *and* revision of a Git repository.
- `https://github.com/NixOS/patchelf/archive/master.tar.gz`: A tarball flake.

## Path-like syntax

Flakes corresponding to a local path can also be referred to by a direct path reference, either `/absolute/path/to/the/flake` or `./relative/path/to/the/flake` (note that the leading `./` is mandatory for relative paths to avoid any ambiguity).

The semantic of such a path is as follows:

- If the directory is part of a Git repository, then the input will be treated as a `git+file:` URL, otherwise it will be treated as a `path: url`;
- If the directory doesn't contain a `flake.nix` file, then Nix will search for such a file

upwards in the file system hierarchy until it finds any of:

1. The Git repository root, or
2. The filesystem root (/), or
3. A folder on a different mount point.

## Examples

- `.` : The flake to which the current directory belongs to.
- `/home/alice/src/patchelf` : A flake in some other directory.

## Flake reference attributes

The following generic flake reference attributes are supported:

- `dir` : The subdirectory of the flake in which `flake.nix` is located. This parameter enables having multiple flakes in a repository or tarball. The default is the root directory of the flake.
- `narHash` : The hash of the NAR serialisation (in SRI format) of the contents of the flake. This is useful for flake types such as tarballs that lack a unique content identifier such as a Git commit hash.

In addition, the following attributes are common to several flake reference types:

- `rev` : A Git or Mercurial commit hash.
- `ref` : A Git or Mercurial branch or tag name.

Finally, some attribute are typically not specified by the user, but can occur in *locked* flake references and are available to Nix code:

- `revCount` : The number of ancestors of the commit `rev` .
- `lastModified` : The timestamp (in seconds since the Unix epoch) of the last modification of this version of the flake. For Git/Mercurial flakes, this is the commit time of commit `rev`, while for tarball flakes, it's the most recent timestamp of any file inside the tarball.

## Types

Currently the `type` attribute can be one of the following:

- **path**: arbitrary local directories, or local Git trees. The required attribute `path` specifies the path of the flake. The URL form is

```
[path:]<path>(\?<params>)?
```

where `path` is an absolute path.

`path` must be a directory in the file system containing a file named `flake.nix`.

`path` generally must be an absolute path. However, on the command line, it can be a relative path (e.g. `.` or `./foo`) which is interpreted as relative to the current directory. In this case, it must start with `.` to avoid ambiguity with registry lookups (e.g. `nixpkgs` is a registry lookup; `./nixpkgs` is a relative path).

- **git**: Git repositories. The location of the repository is specified by the attribute `url`.

They have the URL form

```
git(+http|+https|+ssh|+git|+file|):(//<server>)?<path>(\?<params>)?
```

The `ref` attribute defaults to resolving the `HEAD` reference.

The `rev` attribute must denote a commit that exists in the branch or tag specified by the `ref` attribute, since Nix doesn't do a full clone of the remote repository by default (and the Git protocol doesn't allow fetching a `rev` without a known `ref`). The default is the commit currently pointed to by `ref`.

When `git+file` is used without specifying `ref` or `rev`, files are fetched directly from the local `path` as long as they have been added to the Git repository. If there are uncommitted changes, the reference is treated as dirty and a warning is printed.

For example, the following are valid Git flake references:

- `git+https://example.org/my/repo`
- `git+https://example.org/my/repo?dir=flake1`
- `git+ssh://git@github.com/NixOS/nix?ref=v1.2.3`
- `git://github.com/edolstra/dwarfes?ref=unstable&rev=e486d8d40e626a20e06d792db8cc5ac5aba9a5b4`
- `git+file:///home/my-user/some-repo/some-repo`

- **mercurial**: Mercurial repositories. The URL form is similar to the `git` type, except that the URL schema must be one of `hg+http`, `hg+https`, `hg+ssh` or `hg+file`.

- **tarball**: Tarballs. The location of the tarball is specified by the attribute `url`.

In URL form, the schema must be `tarball+http://`, `tarball+https://` or `tarball+file://`. If the extension corresponds to a known archive format (`.zip`, `.tar`, `.tgz`, `.tar.gz`, `.tar.xz`, `.tar.bz2` or `.tar.zst`), then the `tarball+` can be dropped.

- **file**: Plain files or directory tarballs, either over http(s) or from the local disk.

In URL form, the schema must be `file+http://`, `file+https://` or `file+file://`. If the extension doesn't correspond to a known archive format (as defined by the `tarball` fetcher), then the `file+` prefix can be dropped.

- **github**: A more efficient way to fetch repositories from GitHub. The following attributes are required:

- `owner`: The owner of the repository.
- `repo`: The name of the repository.

These are downloaded as tarball archives, rather than through Git. This is often much faster and uses less disk space since it doesn't require fetching the entire history of the repository. On the other hand, it doesn't allow incremental fetching (but full downloads are often faster than incremental fetches!).

The URL syntax for `github` flakes is:

```
github:<owner>/<repo>(/<rev-or-ref>)?(\?<params>)?
```

`<rev-or-ref>` specifies the name of a branch or tag (`ref`), or a commit hash (`rev`). Note that unlike Git, GitHub allows fetching by commit hash without specifying a branch or tag.

You can also specify `host` as a parameter, to point to a custom GitHub Enterprise server.

Some examples:

- `github:edolstra/dwarfss`
- `github:edolstra/dwarfss/unstable`
- `github:edolstra/dwarfss/d3f2baba8f425779026c6ec04021b2e927f61e31`
- `github:internal/project?host=company-github.example.org`

- **gitlab**: Similar to `github`, is a more efficient way to fetch GitLab repositories. The

following attributes are required:

- `owner` : The owner of the repository.
- `repo` : The name of the repository.

Like `github`, these are downloaded as tarball archives.

The URL syntax for `gitlab` flakes is:

```
gitlab:<owner>/<repo>(/<rev-or-ref>)?(\?<params>)?
```

`<rev-or-ref>` works the same as `github`. Either a branch or tag name (`ref`), or a commit hash (`rev`) can be specified.

Since GitLab allows for self-hosting, you can specify `host` as a parameter, to point to any instances other than `gitlab.com`.

Some examples:

- `gitlab:veloren/veloren`
- `gitlab:veloren/veloren/master`
- `gitlab:veloren/veloren/80a4d7f13492d916e47d6195be23acae8001985a`
- `gitlab:openldap/openldap?host=git.openldap.org`

When accessing a project in a (nested) subgroup, make sure to URL-encode any slashes, i.e. replace `/` with `%2F`:

- `gitlab:veloren%2Fdev/rfc5`
- `sourcehut` : Similar to `github`, is a more efficient way to fetch SourceHut repositories.  
The following attributes are required:
  - `owner` : The owner of the repository (including leading `~`).
  - `repo` : The name of the repository.

Like `github`, these are downloaded as tarball archives.

The URL syntax for `sourcehut` flakes is:

```
sourcehut:<owner>/<repo>(/<rev-or-ref>)?(\?<params>)?
```

`<rev-or-ref>` works the same as `github`. Either a branch or tag name (`ref`), or a commit hash (`rev`) can be specified.

Since SourceHut allows for self-hosting, you can specify `host` as a parameter, to point to any instances other than `git.sr.ht`.

Currently, `ref` name resolution only works for Git repositories. You can refer to Mercurial repositories by simply changing `host` to `hg.sr.ht` (or any other Mercurial instance). With the caveat that you must explicitly specify a commit hash ( `rev` ).

Some examples:

- `sourcehut:~misterio/nix-colors`
- `sourcehut:~misterio/nix-colors/main`
- `sourcehut:~misterio/nix-colors?host=git.example.org`
- `sourcehut:~misterio/nix-colors/182b4b8709b8ffe4e9774a4c5d6877bf6bb9a21c`
- `sourcehut:~misterio/nix-colors/21c1a380a6915d890d408e9f22203436a35bb2de?host=hg.sr.ht`

- `indirect` : Indirections through the flake registry. These have the form

```
[flake:]<flake-id>(/<rev-or-ref>(/rev)?)?
```

These perform a lookup of `<flake-id>` in the flake registry. For example, `nixpkgs` and `nixpkgs/release-20.09` are indirect flake references. The specified `rev` and/or `ref` are merged with the entry in the registry; see [nix registry](#) for details.

## Flake format

As an example, here is a simple `flake.nix` that depends on the Nixpkgs flake and provides a single package (i.e. an [installable](#) derivation):

```
{
 description = "A flake for building Hello World";

 inputs.nixpkgs.url = "github:NixOS/nixpkgs/nixos-20.03";

 outputs = { self, nixpkgs }: {

 packages.x86_64-linux.default =
 # Notice the reference to nixpkgs here.
 with import nixpkgs { system = "x86_64-linux"; };
 stdenv.mkDerivation {
 name = "hello";
 src = self;
 buildPhase = "gcc -o hello ./hello.c";
 installPhase = "mkdir -p $out/bin; install -t $out/bin hello";
 };
 };
}
```

The following attributes are supported in `flake.nix`:

- `description`: A short, one-line description of the flake.
- `inputs`: An attrset specifying the dependencies of the flake (described below).
- `outputs`: A function that, given an attribute set containing the outputs of each of the input flakes keyed by their identifier, yields the Nix values provided by this flake. Thus, in the example above, `inputs.nixpkgs` contains the result of the call to the `outputs` function of the `nixpkgs` flake.

In addition to the outputs of each input, each input in `inputs` also contains some metadata about the inputs. These are:

- `outPath`: The path in the Nix store of the flake's source tree. This way, the attribute set can be passed to `import` as if it was a path, as in the example above (`import nixpkgs`).
- `rev`: The commit hash of the flake's repository, if applicable.
- `revCount`: The number of ancestors of the revision `rev`. This is not available for `github` repositories, since they're fetched as tarballs rather than as Git repositories.
- `lastModifiedDate`: The commit time of the revision `rev`, in the format `%Y%m%d%H%M%S` (e.g. `20181231100934`). Unlike `revCount`, this is available for both

Git and GitHub repositories, so it's useful for generating (hopefully) monotonically increasing version strings.

- `lastModified`: The commit time of the revision `rev` as an integer denoting the number of seconds since 1970.
- `narHash`: The SHA-256 (in SRI format) of the NAR serialization of the flake's source tree.

The value returned by the `outputs` function must be an attribute set. The attributes can have arbitrary values; however, various `nix` subcommands require specific attributes to have a specific value (e.g. `packages.x86_64-linux` must be an attribute set of derivations built for the `x86_64-linux` platform).

- `nixConfig`: a set of `nix.conf` options to be set when evaluating any part of a flake. In the interests of security, only a small set of set of options is allowed to be set without confirmation so long as `accept-flake-config` is not enabled in the global configuration:
  - `bash-prompt`
  - `bash-prompt-prefix`
  - `bash-prompt-suffix`
  - `flake-registry`
  - `commit-lockfile-summary`

## Flake inputs

The attribute `inputs` specifies the dependencies of a flake, as an attrset mapping input names to flake references. For example, the following specifies a dependency on the `nixpkgs` and `import-cargo` repositories:

```
A GitHub repository.
inputs.import-cargo = {
 type = "github";
 owner = "edolstra";
 repo = "import-cargo";
};

An indirection through the flake registry.
inputs.nixpkgs = {
 type = "indirect";
 id = "nixpkgs";
};
```

Alternatively, you can use the URL-like syntax:

```
inputs.import-cargo.url = "github:edolstra/import-cargo";
inputs.nixpkgs.url = "nixpkgs";
```

Each input is fetched, evaluated and passed to the `outputs` function as a set of attributes with the same name as the corresponding input. The special input named `self` refers to the outputs and source tree of *this* flake. Thus, a typical `outputs` function looks like this:

```
outputs = { self, nixpkgs, import-cargo }: {
 ... outputs ...
};
```

It is also possible to omit an input entirely and *only* list it as expected function argument to `outputs`. Thus,

```
outputs = { self, nixpkgs }: ...;
```

without an `inputs.nixpkgs` attribute is equivalent to

```
inputs.nixpkgs = {
 type = "indirect";
 id = "nixpkgs";
};
```

Repositories that don't contain a `flake.nix` can also be used as inputs, by setting the input's `flake` attribute to `false`:

```
inputs.grcov = {
 type = "github";
 owner = "mozilla";
 repo = "grcov";
 flake = false;
};

outputs = { self, nixpkgs, grcov }: {
 packages.x86_64-linux.grcov = stdenv.mkDerivation {
 src = grcov;
 ...
 };
};
```

Transitive inputs can be overridden from a `flake.nix` file. For example, the following overrides the `nixpkgs` input of the `nixops` input:

```
inputs.nixops.inputs.nixpkgs = {
 type = "github";
 owner = "my-org";
 repo = "nixpkgs";
};
```

It is also possible to "inherit" an input from another input. This is useful to minimize flake dependencies. For example, the following sets the `nixpkgs` input of the top-level flake to be equal to the `nixpkgs` input of the `dwarfffs` input of the top-level flake:

```
inputs.nixpkgs.follows = "dwarfffs/nixpkgs";
```

The value of the `follows` attribute is a `/`-separated sequence of input names denoting the path of inputs to be followed from the root flake.

Overrides and `follows` can be combined, e.g.

```
inputs.nixops.inputs.nixpkgs.follows = "dwarfffs/nixpkgs";
```

sets the `nixpkgs` input of `nixops` to be the same as the `nixpkgs` input of `dwarfffs`. It is worth noting, however, that it is generally not useful to eliminate transitive `nixpkgs` flake inputs in this way. Most flakes provide their functionality through Nixpkgs overlays or NixOS modules, which are composed into the top-level flake's `nixpkgs` input; so their own `nixpkgs` input is usually irrelevant.

## Lock files

Inputs specified in `flake.nix` are typically "unlocked" in the sense that they don't specify an exact revision. To ensure reproducibility, Nix will automatically generate and use a *lock file* called `flake.lock` in the flake's directory. The lock file contains a graph structure isomorphic to the graph of dependencies of the root flake. Each node in the graph (except the root node) maps the (usually) unlocked input specifications in `flake.nix` to locked input specifications. Each node also contains some metadata, such as the dependencies (outgoing edges) of the node.

For example, if `flake.nix` has the inputs in the example above, then the resulting lock file might be:

```
{
 "version": 7,
 "root": "n1",
 "nodes": {
 "n1": {
 "inputs": {
 "nixpkgs": "n2",
 "import-cargo": "n3",
 "grcov": "n4"
 }
 },
 "n2": {
 "inputs": {},
 "locked": {
 "owner": "edolstra",
 "repo": "nixpkgs",
 "rev": "7f8d4b088e2df7fdb6b513bc2d6941f1d422a013",
 "type": "github",
 "lastModified": 1580555482,
 "narHash": "sha256-0npEWzNx/F/AU4KlqBXM2s5PwvfI5/BS6xQrPvkF5t08="
 },
 "original": {
 "id": "nixpkgs",
 "type": "indirect"
 }
 },
 "n3": {
 "inputs": {},
 "locked": {
 "owner": "edolstra",
 "repo": "import-cargo",
 "rev": "8abf7b3a8cbe1c8a885391f826357a74d382a422",
 "type": "github",
 "lastModified": 1567183309,
 "narHash": "sha256-wIXWOpX9rRjK5NDsL6WzuuBJl2R0kUCnlpZUrASykSc="
 },
 "original": {
 "owner": "edolstra",
 "repo": "import-cargo",
 "type": "github"
 }
 },
 "n4": {
 "inputs": {},
 "locked": {
 "owner": "mozilla",
 "repo": "grcov",
 "rev": "989a84bb29e95e392589c4e73c29189fd69a1d4e",
 "type": "github",
 "lastModified": 1580729070,
 "narHash": "sha256-235uMxYlHxJ5y92EXZWAYEsEb6mm+b069GAd+BOIOxI="
 },
 },
 },
}
```

```
"original": {
 "owner": "mozilla",
 "repo": "grcov",
 "type": "github"
},
"flake": false
}
}
}
```

This graph has 4 nodes: the root flake, and its 3 dependencies. The nodes have arbitrary labels (e.g. `n1`). The label of the root node of the graph is specified by the `root` attribute. Nodes contain the following fields:

- `inputs`: The dependencies of this node, as a mapping from input names (e.g. `nixpkgs`) to node labels (e.g. `n2`).
- `original`: The original input specification from `flake.lock`, as a set of `builtins.fetchTree` arguments.
- `locked`: The locked input specification, as a set of `builtins.fetchTree` arguments. Thus, in the example above, when we build this flake, the input `nixpkgs` is mapped to revision `7f8d4b088e2df7fdb6b513bc2d6941f1d422a013` of the `edolstra/nixpkgs` repository on GitHub.

It also includes the attribute `narHash`, specifying the expected contents of the tree in the Nix store (as computed by `nix hash-path`), and may include input-type-specific attributes such as the `lastModified` or `revCount`. The main reason for these attributes is to allow flake inputs to be substituted from a binary cache: `narHash` allows the store path to be computed, while the other attributes are necessary because they provide information not stored in the store path.

- `flake`: A Boolean denoting whether this is a flake or non-flake dependency. Corresponds to the `flake` attribute in the `inputs` attribute in `flake.nix`.

The `original` and `locked` attributes are omitted for the root node. This is because we cannot record the commit hash or content hash of the root flake, since modifying `flake.lock` will invalidate these.

The graph representation of lock files allows circular dependencies between flakes. For example, here are two flakes that reference each other:

```
{
 inputs.b = ... location of flake B ...;
 # Tell the 'b' flake not to fetch 'a' again, to ensure its 'a' is
 # *this* 'a'.
 inputs.b.inputs.a.follows = "";
 outputs = { self, b }: {
 foo = 123 + b.bar;
 xyzzy = 1000;
 };
}
```

and

```
{
 inputs.a = ... location of flake A ...;
 inputs.a.inputs.b.follows = "";
 outputs = { self, a }: {
 bar = 456 + a.xyzzy;
 };
}
```

Lock files transitively lock direct as well as indirect dependencies. That is, if a lock file exists and is up to date, Nix will not look at the lock files of dependencies. However, lock file generation itself *does* use the lock files of dependencies by default.

---

**Warning**

This program is **experimental** and its interface is subject to change.

---

# Name

`nix flake archive` - copy a flake and all its inputs to a store

# Synopsis

`nix flake archive [option...] flake-url`

# Examples

- Copy the `dwarfss` flake and its dependencies to a binary cache:

```
nix flake archive --to file:///tmp/my-cache dwarfss
```

- Fetch the `dwarfss` flake and its dependencies to the local Nix store:

```
nix flake archive dwarfss
```

- Print the store paths of the flake sources of NixOps without fetching them:

```
nix flake archive --json --dry-run nixops
```

# Description

FIXME

# Options

- `--dry-run` Show what this command would do without doing it.
- `--json` Produce output in JSON format, suitable for consumption by another program.
- `--to store-uri` URI of the destination Nix store

## Common evaluation options:

- `--arg name expr` Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr name string` Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store store-url` The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include / -I path` Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location.

The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.

- **--output-lock-file** *flake-lock-path* Write the given lock file instead of `flake.lock` within the top-level flake.
- **--override-input** *input-path flake-url* Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies **--no-write-lock-file**.
- **--recreate-lock-file** Recreate the flake's lock file from scratch.
- **--reference-lock-file** *flake-lock-path* Read the given lock file instead of `flake.lock` within the top-level flake.
- **--update-input** *input-path* Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- **--debug** Set the logging verbosity level to 'debug'.
- **--log-format** *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- **--print-build-logs** / **-L** Print full build logs on standard error.
- **--quiet** Decrease the logging verbosity level.
- **--verbose** / **-v** Increase the logging verbosity level.

## Miscellaneous global options:

- **--help** Show usage information.
- **--offline** Disable substituters and consider all previously downloaded files up-to-date.
- **--option** *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- **--refresh** Consider all previously downloaded files out-of-date.
- **--repair** During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- **--version** Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.

- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors value` Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls value` Append to the `ignored-acls` setting.
- `--extra-nix-path value` Append to the `nix-path` setting.
- `--extra-platforms value` Set the `extra-platforms` setting.
- `--extra-plugin-files value` Append to the `plugin-files` setting.
- `--extra-sandbox-paths value` Append to the `sandbox-paths` setting.
- `--extra-secret-key-files value` Append to the `secret-key-files` setting.
- `--extra-substituters value` Append to the `substituters` setting.
- `--extra-system-features value` Append to the `system-features` setting.
- `--extra-trusted-public-keys value` Append to the `trusted-public-keys` setting.

- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.

- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.

- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.

- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.

- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.
- `--start-id value` Set the `start-id` setting.
- `--store value` Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix flake check` - check whether the flake evaluates and run its tests

# Synopsis

`nix flake check [option...] flake-url`

# Examples

- Evaluate the flake in the current directory, and build its checks:

```
nix flake check
```

- Verify that the `patchelf` flake evaluates, but don't build its checks:

```
nix flake check --no-build github:NixOS/patchelf
```

# Description

This command verifies that the flake specified by flake reference *flake-url* can be evaluated successfully (as detailed below), and that the derivations specified by the flake's `checks` output can be built successfully.

If the `keep-going` option is set to `true`, Nix will keep evaluating as much as it can and report the errors as it encounters them. Otherwise it will stop at the first error.

# Evaluation checks

The following flake output attributes must be derivations:

- `checks.system.name`
- `defaultPackage.system`
- `devShell.system`
- `devShells.system.name`
- `nixosConfigurations.name.config.system.build.toplevel`
- `packages.system.name`

The following flake output attributes must be [app definitions](#):

- `apps.system.name`
- `defaultApp.system`

The following flake output attributes must be [template definitions](#):

- `defaultTemplate`
- `templates.name`

The following flake output attributes must be *Nixpkgs overlays*:

- `overlay`
- `overlays.name`

The following flake output attributes must be *NixOS modules*:

- `nixosModule`
- `nixosModules.name`

The following flake output attributes must be [bundlers](#):

- `bundlers.name`
- `defaultBundler`

In addition, the `hydraJobs` output is evaluated in the same way as Hydra's `hydra-eval-jobs` (i.e. as a arbitrarily deeply nested attribute set of derivations). Similarly, the `legacyPackages.system` output is evaluated like `nix-env --query --available`.

# Options

- `--all-systems` Check the outputs for all systems.
- `--no-build` Do not build checks.

## Common evaluation options:

- `--arg` *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr` *name string* Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store` *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include` / `-I` *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using [paths](#) enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
```

```
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock`

within the top-level flake.

- `--override-input` *input-path flake-url* Override a specific flake input (e.g. `dwarfss/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file` *flake-lock-path* Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input` *input-path* Update a specific flake input (ignoring its previous entry in the lock file).

### Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

### Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

### Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.

- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.

- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.

- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.
- `--max-silent-time` *value* Set the `max-silent-time` setting.

- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.

- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.

- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.

- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.
- `--start-id value` Set the `start-id` setting.
- `--store value` Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix flake clone` - clone flake repository

# Synopsis

`nix flake clone [option...] flake-url`

# Examples

- Check out the source code of the `dwarfss` flake and build it:

```
nix flake clone dwarfss --dest dwarfss
cd dwarfss
nix build
```

# Description

This command performs a Git or Mercurial clone of the repository containing the source code of the flake *flake-url*.

# Options

- `--dest` / `-f` *path* Clone the flake to path *dest*.

## Common evaluation options:

- `--arg` *name expr* Pass the value *expr* as the argument *name* to Nix functions.

- **--argstr** *name string* Pass the string *string* as the argument *name* to Nix functions.
- **--debugger** Start an interactive environment if evaluation fails.
- **--eval-store** *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- **--impure** Allow access to mutable paths and repositories.
- **--include** / **-I** *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for](#)

`nix-channel`) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.

- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.

- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.

- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.

- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.
- `--max-silent-time` *value* Set the `max-silent-time` setting.
- `--max-substitution-jobs` *value* Set the `max-substitution-jobs` setting.
- `--min-free` *value* Set the `min-free` setting.
- `--min-free-check-interval` *value* Set the `min-free-check-interval` setting.

- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.

- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.

- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.
- `--start-id value` Set the `start-id` setting.

- `--store value` Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix flake info` - show flake metadata

## Synopsis

`nix flake info [option...] flake-url`

# Examples

- Show what `nixpkgs` resolves to:

```
nix flake metadata nixpkgs
Resolved URL: github:edolstra/dwarfss
Locked URL: github:edolstra/dwarfss
/f691e2c991e75edb22836f1dbe632c40324215c5
Description: A filesystem that fetches DWARF debug info from the Internet
on demand
Path: /nix/store/769s05vjydmc2lcf6b02az28wsa9ixh1-source
Revision: f691e2c991e75edb22836f1dbe632c40324215c5
Last modified: 2021-01-21 15:41:26
Inputs:
├─nix: github:NixOS/nix/6254b1f5d298ff73127d7b0f0da48f142bdc753c
| └─lowdown-src: github:kristapsdz/lowdown
/1705b4a26fbf065d9574dce47a94e8c7c79e052f
| └─nixpkgs: github:NixOS/nixpkgs
/ad0d20345219790533ebe06571f82ed6b034db31
└─nixpkgs follows input 'nix/nixpkgs'
```

- Show information about `dwarfss` in JSON format:

```
nix flake metadata dwarffs --json | jq .

{
 "description": "A filesystem that fetches DWARF debug info from the Internet on demand",
 "lastModified": 1597153508,
 "locked": {
 "lastModified": 1597153508,
 "narHash": "sha256-VHg3MYVgQ12LeRSU2PSoDeKlSPD8PYyEFxxwkVVDRd0=",
 "owner": "edolstra",
 "repo": "dwarfss",
 "rev": "d181d714fd36eb06f4992a1997cd5601e26db8f5",
 "type": "github"
 },
 "locks": { ... },
 "original": {
 "id": "dwarfss",
 "type": "indirect"
 },
 "originalUrl": "flake:dwarfss",
 "path": "/nix/store/hang3792qwdmm2n0d9nsrs5n6bsws6kv-source",
 "resolved": {
 "owner": "edolstra",
 "repo": "dwarfss",
 "type": "github"
 },
 "resolvedUrl": "github:edolstra/dwarfss",
 "revision": "d181d714fd36eb06f4992a1997cd5601e26db8f5",
 "url": "github:edolstra/dwarfss/d181d714fd36eb06f4992a1997cd5601e26db8f5"
}
```

# Description

This command shows information about the flake specified by the flake reference *flake-url*. It resolves the flake reference using the [flake registry](#), fetches it, and prints some meta data. This includes:

- **Resolved URL** : If *flake-url* is a flake identifier, then this is the flake reference that specifies its actual location, looked up in the flake registry.

- **Locked URL**: A flake reference that contains a commit or content hash and thus uniquely identifies a specific flake version.
- **Description**: A one-line description of the flake, taken from the `description` field in `flake.nix`.
- **Path**: The store path containing the source code of the flake.
- **Revision**: The Git or Mercurial commit hash of the locked flake.
- **Revisions**: The number of ancestors of the Git or Mercurial commit of the locked flake. Note that this is not available for `github` flakes.
- **Last modified**: For Git or Mercurial flakes, this is the commit time of the commit of the locked flake; for tarball flakes, it's the most recent timestamp of any file inside the tarball.
- **Inputs**: The flake inputs with their corresponding lock file entries.

With `--json`, the output is a JSON object with the following fields:

- **original** and **originalUrl**: The flake reference specified by the user (*flake-url*) in attribute set and URL representation.
- **resolved** and **resolvedUrl**: The resolved flake reference (see above) in attribute set and URL representation.
- **locked** and **lockedUrl**: The locked flake reference (see above) in attribute set and URL representation.
- **description**: See **Description** above.
- **path**: See **Path** above.
- **revision**: See **Revision** above.
- **revCount**: See **Revisions** above.
- **lastModified**: See **Last modified** above.
- **locks**: The contents of `flake.lock`.

# Options

- **--json** Produce output in JSON format, suitable for consumption by another program.

## Common evaluation options:

- **--arg** *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- **--argstr** *name string* Pass the string *string* as the argument *name* to Nix functions.
- **--debugger** Start an interactive environment if evaluation fails.
- **--eval-store** *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- **--impure** Allow access to mutable paths and repositories.
- **--include** / **-I** *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g.

`dwarffs/nixpkgs`). This implies `--no-write-lock-file`.

- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.

- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.

- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.

- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.
- `--max-silent-time` *value* Set the `max-silent-time` setting.
- `--max-substitution-jobs` *value* Set the `max-substitution-jobs` setting.

- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.

- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.

- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.

- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix flake init` - create a flake in the current directory from a template

# Synopsis

`nix flake init [option...]`

# Examples

- Create a flake using the default template:

```
nix flake init
```

- List available templates:

```
nix flake show templates
```

- Create a flake from a specific template:

```
nix flake init -t templates#simpleContainer
```

# Description

This command creates a flake in the current directory by copying the files of a template. It will not overwrite existing files. The default template is `templates#templates.default`, but this can be overridden using `-t`.

# Template definitions

A flake can declare templates through its `templates` output attribute. A template has two attributes:

- `description`: A one-line description of the template, in CommonMark syntax.
- `path`: The path of the directory to be copied.
- `welcomeText`: A block of markdown text to display when a user initializes a new flake based on this template.

Here is an example:

```
outputs = { self }: {

 templates.rust = {
 path = ./rust;
 description = "A simple Rust/Cargo project";
 welcomeText = ''
 # Simple Rust/Cargo Template
 ## Intended usage
 The intended usage of this flake is...

 ## More info
 - [Rust language](https://www.rust-lang.org/)
 - [Rust on the NixOS Wiki](https://nixos.wiki/wiki/Rust)
 -
 '';
 };

 templates.default = self.templates.rust;
}
```

# Options

- `--template` / `-t template` The template to use.

## Common evaluation options:

- `--arg name expr` Pass the value `expr` as the argument `name` to Nix functions.
- `--argstr name string` Pass the string `string` as the argument `name` to Nix functions.

- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store store-url` The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include / -I path` Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.

- **--repair** During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- **--version** Show version information.

## Options to override configuration settings:

- **--accept-flake-config** Enable the `accept-flake-config` setting.
- **--access-tokens** *value* Set the `access-tokens` setting.
- **--allow-dirty** Enable the `allow-dirty` setting.
- **--allow-import-from-derivation** Enable the `allow-import-from-derivation` setting.
- **--allow-new-privileges** Enable the `allow-new-privileges` setting.
- **--allow-symlinked-store** Enable the `allow-symlinked-store` setting.
- **--allow-unsafe-native-code-during-evaluation** Enable the `allow-unsafe-native-code-during-evaluation` setting.
- **--allowed-impure-host-deps** *value* Set the `allowed-impure-host-deps` setting.
- **--allowed-uris** *value* Set the `allowed-uris` setting.
- **--allowed-users** *value* Set the `allowed-users` setting.
- **--auto-allocate-uids** Enable the `auto-allocate-uids` setting.
- **--auto-optimise-store** Enable the `auto-optimise-store` setting.
- **--bash-prompt** *value* Set the `bash-prompt` setting.
- **--bash-prompt-prefix** *value* Set the `bash-prompt-prefix` setting.
- **--bash-prompt-suffix** *value* Set the `bash-prompt-suffix` setting.
- **--build-hook** *value* Set the `build-hook` setting.
- **--build-poll-interval** *value* Set the `build-poll-interval` setting.
- **--build-users-group** *value* Set the `build-users-group` setting.
- **--builders** *value* Set the `builders` setting.

- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors value` Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls value` Append to the `ignored-acls` setting.
- `--extra-nix-path value` Append to the `nix-path` setting.
- `--extra-platforms value` Set the `extra-platforms` setting.
- `--extra-plugin-files value` Append to the `plugin-files` setting.
- `--extra-sandbox-paths value` Append to the `sandbox-paths` setting.

- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.

- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.

- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.

- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.

- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.

- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix flake lock` - create missing lock file entries

# Synopsis

`nix flake lock [option...] flake-url`

# Examples

- Update the `nixpkgs` and `nix` inputs of the flake in the current directory:

```
nix flake lock --update-input nixpkgs --update-input nix
* Updated 'nix': 'github:NixOS/nix
/9fab14adbc3810d5cc1f88672fde1eee4358405c' -> 'github:NixOS/nix
/8927cba62f5afb33b01016d5c4f7f8b7d0adde3c'
* Updated 'nixpkgs': 'github:NixOS/nixpkgs
/3d2d8f281a27d466fa54b469b5993f7dde198375' -> 'github:NixOS/nixpkgs
/a3a3dda3bacf61e8a39258a0ed9c924eeeca8e293'
```

# Description

This command updates the lock file of a flake (`flake.lock`) so that it contains a lock for every flake input specified in `flake.nix`. Existing lock file entries are not updated unless required by a flag such as `--update-input`.

Note that every command that operates on a flake will also update the lock file if needed, and supports the same flags. Therefore,

```
nix flake lock --update-input nixpkgs
nix build
```

is equivalent to:

```
nix build --update-input nixpkgs
```

Thus, this command is only useful if you want to update the lock file separately from any other action such as building.

# Options

## Common evaluation options:

- `--arg` *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr` *name string* Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store` *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include` / `-I` *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for nix-channel](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).

- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.

- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.

- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.

- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.

- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.

- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.

- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.

- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix flake metadata` - show flake metadata

## Synopsis

`nix flake metadata [option...] flake-url`

# Examples

- Show what `nixpkgs` resolves to:

```
nix flake metadata nixpkgs
Resolved URL: github:edolstra/dwarfss
Locked URL: github:edolstra/dwarfss
/f691e2c991e75edb22836f1dbe632c40324215c5
Description: A filesystem that fetches DWARF debug info from the Internet
on demand
Path: /nix/store/769s05vjydmc2lcf6b02az28wsa9ixh1-source
Revision: f691e2c991e75edb22836f1dbe632c40324215c5
Last modified: 2021-01-21 15:41:26
Inputs:
├─nix: github:NixOS/nix/6254b1f5d298ff73127d7b0f0da48f142bdc753c
| └─lowdown-src: github:kristapsdz/lowdown
/1705b4a26fbf065d9574dce47a94e8c7c79e052f
| └─nixpkgs: github:NixOS/nixpkgs
/ad0d20345219790533ebe06571f82ed6b034db31
└─nixpkgs follows input 'nix/nixpkgs'
```

- Show information about `dwarfss` in JSON format:

```
nix flake metadata dwarffs --json | jq .

{
 "description": "A filesystem that fetches DWARF debug info from the Internet on demand",
 "lastModified": 1597153508,
 "locked": {
 "lastModified": 1597153508,
 "narHash": "sha256-VHg3MYVgQ12LeRSU2PSoDeKlSPD8PYyEFxxwkVVDRd0=",
 "owner": "edolstra",
 "repo": "dwarfss",
 "rev": "d181d714fd36eb06f4992a1997cd5601e26db8f5",
 "type": "github"
 },
 "locks": { ... },
 "original": {
 "id": "dwarfss",
 "type": "indirect"
 },
 "originalUrl": "flake:dwarfss",
 "path": "/nix/store/hang3792qwdmm2n0d9nsrs5n6bsws6kv-source",
 "resolved": {
 "owner": "edolstra",
 "repo": "dwarfss",
 "type": "github"
 },
 "resolvedUrl": "github:edolstra/dwarfss",
 "revision": "d181d714fd36eb06f4992a1997cd5601e26db8f5",
 "url": "github:edolstra/dwarfss/d181d714fd36eb06f4992a1997cd5601e26db8f5"
}
```

# Description

This command shows information about the flake specified by the flake reference *flake-url*. It resolves the flake reference using the [flake registry](#), fetches it, and prints some meta data. This includes:

- **Resolved URL** : If *flake-url* is a flake identifier, then this is the flake reference that specifies its actual location, looked up in the flake registry.

- **Locked URL**: A flake reference that contains a commit or content hash and thus uniquely identifies a specific flake version.
- **Description**: A one-line description of the flake, taken from the `description` field in `flake.nix`.
- **Path**: The store path containing the source code of the flake.
- **Revision**: The Git or Mercurial commit hash of the locked flake.
- **Revisions**: The number of ancestors of the Git or Mercurial commit of the locked flake. Note that this is not available for `github` flakes.
- **Last modified**: For Git or Mercurial flakes, this is the commit time of the commit of the locked flake; for tarball flakes, it's the most recent timestamp of any file inside the tarball.
- **Inputs**: The flake inputs with their corresponding lock file entries.

With `--json`, the output is a JSON object with the following fields:

- **original** and **originalUrl**: The flake reference specified by the user (*flake-url*) in attribute set and URL representation.
- **resolved** and **resolvedUrl**: The resolved flake reference (see above) in attribute set and URL representation.
- **locked** and **lockedUrl**: The locked flake reference (see above) in attribute set and URL representation.
- **description**: See **Description** above.
- **path**: See **Path** above.
- **revision**: See **Revision** above.
- **revCount**: See **Revisions** above.
- **lastModified**: See **Last modified** above.
- **locks**: The contents of `flake.lock`.

# Options

- **--json** Produce output in JSON format, suitable for consumption by another program.

## Common evaluation options:

- **--arg** *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- **--argstr** *name string* Pass the string *string* as the argument *name* to Nix functions.
- **--debugger** Start an interactive environment if evaluation fails.
- **--eval-store** *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- **--impure** Allow access to mutable paths and repositories.
- **--include** / **-I** *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g.

`dwarffs/nixpkgs`). This implies `--no-write-lock-file`.

- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.

- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.

- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.

- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.
- `--max-silent-time` *value* Set the `max-silent-time` setting.
- `--max-substitution-jobs` *value* Set the `max-substitution-jobs` setting.

- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.

- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.

- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.

- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix flake new` - create a flake in the specified directory from a template

# Synopsis

`nix flake new [option...] dest-dir`

# Examples

- Create a flake using the default template in the directory `hello`:

```
nix flake new hello
```

- List available templates:

```
nix flake show templates
```

- Create a flake from a specific template in the directory `hello`:

```
nix flake new hello -t templates#trivial
```

# Description

This command creates a flake in the directory `dest-dir`, which must not already exist. It's equivalent to:

```
mkdir dest-dir
cd dest-dir
nix flake init
```

# Options

- `--template` / `-t template` The template to use.

## Common evaluation options:

- `--arg` *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr` *name string* Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store` *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include` / `-I path` Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.

- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps value` Set the `allowed-impure-host-deps` setting.
- `--allowed-uris value` Set the `allowed-uris` setting.
- `--allowed-users value` Set the `allowed-users` setting.

- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.

- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.

- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.
- `--max-silent-time` *value* Set the `max-silent-time` setting.
- `--max-substitution-jobs` *value* Set the `max-substitution-jobs` setting.
- `--min-free` *value* Set the `min-free` setting.
- `--min-free-check-interval` *value* Set the `min-free-check-interval` setting.
- `--nar-buffer-size` *value* Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl` *value* Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl` *value* Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file` *value* Set the `netrc-file` setting.
- `--nix-path` *value* Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.

- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.

- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.

- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.

- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

**nix flake prefetch** - download the source tree denoted by a flake reference into the Nix store

# Synopsis

**nix flake prefetch** [*option...*] *flake-url*

# Examples

- Download a tarball and unpack it:

```
nix flake prefetch https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.10.5.tar.xz
Downloaded 'https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.10.5.tar.xz?narHash=sha256-3XYHZANT6AFBV0BqegkAZHbba6oeDkIUCDwbATLMhAY='
to '/nix/store/sl5vvk8mb4ma1sjyy03kwpvkz50hd22d-source' (hash
'sha256-3XYHZANT6AFBV0BqegkAZHbba6oeDkIUCDwbATLMhAY=').
```

- Download the **dwarf** flake (looked up in the flake registry):

```
nix flake prefetch dwarf --json
>{"hash":"sha256-VHg3MYVgQ12LeRSU2PSoDeKlSPD8PYYEFxxwkVVDRd0=",
 "storePath":"/nix/store/hang3792qwdmm2n0d9nsrs5n6bsws6kv-source"}
```

# Description

This command downloads the source tree denoted by flake reference *flake-url*. Note that

this does not need to be a flake (i.e. it does not have to contain a `flake.nix` file).

# Options

- `--json` Produce output in JSON format, suitable for consumption by another program.

## Common evaluation options:

- `--arg` *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr` *name string* Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store` *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include` / `-I` *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

### Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.

- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file` *flake-lock-path* Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input` *input-path flake-url* Override a specific flake input (e.g. `dwarfss/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file` *flake-lock-path* Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input` *input-path* Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.

- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.

- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors value` Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls value` Append to the `ignored-acls` setting.
- `--extra-nix-path value` Append to the `nix-path` setting.
- `--extra-platforms value` Set the `extra-platforms` setting.
- `--extra-plugin-files value` Append to the `plugin-files` setting.
- `--extra-sandbox-paths value` Append to the `sandbox-paths` setting.
- `--extra-secret-key-files value` Append to the `secret-key-files` setting.
- `--extra-substituters value` Append to the `substituters` setting.

- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.

- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.

- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.

- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.

- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.
- `--start-id value` Set the `start-id` setting.
- `--store value` Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.

- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

**Warning**

This program is **experimental** and its interface is subject to change.

---

# Name

`nix flake show` - show the outputs provided by a flake

# Synopsis

`nix flake show [option...] flake-url`

# Examples

- Show the output attributes provided by the `patchelf` flake:

```
github:NixOS/patchelf/f34751b88bd07d7f44f5cd3200fb4122bf916c7e
 ├── checks
 │ ├── aarch64-linux
 │ │ └── build: derivation 'patchelf-0.12.20201207.f34751b'
 │ ├── i686-linux
 │ │ └── build: derivation 'patchelf-0.12.20201207.f34751b'
 │ └── x86_64-linux
 │ └── build: derivation 'patchelf-0.12.20201207.f34751b'
 ├── packages
 │ ├── aarch64-linux
 │ │ └── default: package 'patchelf-0.12.20201207.f34751b'
 │ ├── i686-linux
 │ │ └── default: package 'patchelf-0.12.20201207.f34751b'
 │ └── x86_64-linux
 │ └── default: package 'patchelf-0.12.20201207.f34751b'
 ├── hydraJobs
 │ ├── build
 │ │ ├── aarch64-linux: derivation 'patchelf-0.12.20201207.f34751b'
 │ │ ├── i686-linux: derivation 'patchelf-0.12.20201207.f34751b'
 │ │ └── x86_64-linux: derivation 'patchelf-0.12.20201207.f34751b'
 │ ├── coverage: derivation 'patchelf-coverage-0.12.20201207.f34751b'
 │ ├── release: derivation 'patchelf-0.12.20201207.f34751b'
 │ └── tarball: derivation 'patchelf-tarball-0.12.20201207.f34751b'
 └── overlay: Nixpkgs overlay
```

# Description

This command shows the output attributes provided by the flake specified by flake reference `flake-url`. These are the top-level attributes in the `outputs` of the flake, as well as lower-level attributes for some standard outputs (e.g. `packages` or `checks`).

With `--json`, the output is in a JSON representation suitable for automatic processing by other tools.

# Options

- `--all-systems` Show the contents of outputs for all systems.

- `--json` Produce output in JSON format, suitable for consumption by another program.
- `--legacy` Show the contents of the `legacyPackages` output.

## Common evaluation options:

- `--arg` *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr` *name string* Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store` *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include` / `-I` *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock`

within the top-level flake.

- `--override-input` *input-path flake-url* Override a specific flake input (e.g. `dwarfss/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file` *flake-lock-path* Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input` *input-path* Update a specific flake input (ignoring its previous entry in the lock file).

### Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

### Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

### Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.

- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.

- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.

- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.
- `--max-silent-time` *value* Set the `max-silent-time` setting.

- `--max-substitution-jobs` *value* Set the `max-substitution-jobs` setting.
- `--min-free` *value* Set the `min-free` setting.
- `--min-free-check-interval` *value* Set the `min-free-check-interval` setting.
- `--nar-buffer-size` *value* Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl` *value* Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl` *value* Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file` *value* Set the `netrc-file` setting.
- `--nix-path` *value* Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.

- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.

- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.

- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.
- `--start-id value` Set the `start-id` setting.
- `--store value` Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix flake update` - update flake lock file

# Synopsis

`nix flake update [option...] flake-url`

# Examples

- Recreate the lock file (i.e. update all inputs) and commit the new lock file:

```
nix flake update --commit-lock-file
* Updated 'nix': 'github:NixOS/nix
/9fab14adbc3810d5cc1f88672fde1eee4358405c' -> 'github:NixOS/nix
/8927cba62f5afb33b01016d5c4f7f8b7d0adde3c'
* Updated 'nixpkgs': 'github:NixOS/nixpkgs
/3d2d8f281a27d466fa54b469b5993f7dde198375' -> 'github:NixOS/nixpkgs
/a3a3dda3bacf61e8a39258a0ed9c924eeeca8e293'

...
warning: committed new revision '158bcd9d6cc08ab859c0810186c1beebc982aad'
```

# Description

This command recreates the lock file of a flake (`flake.lock`), thus updating the lock for every unlocked input (like `nixpkgs`) to its current version. This is equivalent to passing `--recreate-lock-file` to any command that operates on a flake. That is,

```
nix flake update
nix build
```

is equivalent to:

```
nix build --recreate-lock-file
```

# Options

## Common evaluation options:

- `--arg` *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr` *name string* Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store` *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include` / `-I` *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>` ).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for nix-channel](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarfss/nixpkgs`). This implies `--no-write-lock-file`.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.

- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors value` Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls value` Append to the `ignored-acls` setting.
- `--extra-nix-path value` Append to the `nix-path` setting.
- `--extra-platforms value` Set the `extra-platforms` setting.
- `--extra-plugin-files value` Append to the `plugin-files` setting.
- `--extra-sandbox-paths value` Append to the `sandbox-paths` setting.
- `--extra-secret-key-files value` Append to the `secret-key-files` setting.
- `--extra-substituters value` Append to the `substituters` setting.
- `--extra-system-features value` Append to the `system-features` setting.

- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.

- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.

- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.

- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.

- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.
- `--start-id value` Set the `start-id` setting.
- `--store value` Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.

- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix fmt` - reformat your code in the standard style

# Synopsis

`nix fmt [option...] args...`

# Examples

With `nixpkgs-fmt`:

```
flake.nix
{
 outputs = { nixpkgs, self }: {
 formatter.x86_64-linux = nixpkgs.legacyPackages.x86_64-linux.nixpkgs-fmt;
 };
}
```

- Format the current flake: `$ nix fmt`
- Format a specific folder or file: `$ nix fmt ./folder ./file.nix`

With `nixfmt`:

```
flake.nix
{
 outputs = { nixpkgs, self }: {
 formatter.x86_64-linux = nixpkgs.legacyPackages.x86_64-linux.nixfmt;
 };
}
```

- Format specific files: `$ nix fmt ./file1.nix ./file2.nix`

With Alejandra:

```
flake.nix
{
 outputs = { nixpkgs, self }: {
 formatter.x86_64-linux = nixpkgs.legacyPackages.x86_64-linux.alejandra;
 };
}
```

- Format the current flake: `$ nix fmt`
- Format a specific folder or file: `$ nix fmt ./folder ./file.nix`

# Description

`nix fmt` will rewrite all Nix files (\*.nix) to a canonical format using the formatter specified in your flake.

# Options

## Common evaluation options:

- `--arg` *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr` *name string* Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store` *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include` / `-I` *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.

- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options that change the interpretation of `installables`:

- `--expr` *expr* Interpret `installables` as attribute paths relative to the Nix expression *expr*.
- `--file` / `-f` *file* Interpret `installables` as attribute paths relative to the Nix expression stored in *file*. If *file* is the character -, then a Nix expression will be read from standard input. Implies `--impure`.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.

- `--allowed-uris value` Set the `allowed-uris` setting.
- `--allowed-users value` Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt value` Set the `bash-prompt` setting.
- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.
- `--build-hook value` Set the `build-hook` setting.
- `--build-poll-interval value` Set the `build-poll-interval` setting.
- `--build-users-group value` Set the `build-users-group` setting.
- `--builders value` Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.

- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.

- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.
- `--max-silent-time` *value* Set the `max-silent-time` setting.
- `--max-substitution-jobs` *value* Set the `max-substitution-jobs` setting.
- `--min-free` *value* Set the `min-free` setting.
- `--min-free-check-interval` *value* Set the `min-free-check-interval` setting.
- `--nar-buffer-size` *value* Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl` *value* Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl` *value* Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file` *value* Set the `netrc-file` setting.

- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.

- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.

- `--pre-build-hook` *value* Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.

- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

**Warning**

This program is **experimental** and its interface is subject to change.

---

# Name

`nix hash` - compute and convert cryptographic hashes

# Synopsis

`nix hash [option...] subcommand`

where *subcommand* is one of the following:

- `nix hash file` - print cryptographic hash of a regular file
- `nix hash path` - print cryptographic hash of the NAR serialisation of a path
- `nix hash to-base16` - convert a hash to base-16 representation
- `nix hash to-base32` - convert a hash to base-32 representation
- `nix hash to-base64` - convert a hash to base-64 representation
- `nix hash to-sri` - convert a hash to SRI representation

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix hash file` - print cryptographic hash of a regular file

## Synopsis

`nix hash file [option...] paths...`

# Options

- `--base16` Print the hash in base-16 format.
- `--base32` Print the hash in base-32 (Nix-specific) format.
- `--base64` Print the hash in base-64 format.
- `--sri` Print the hash in SRI format.
- `--type hash-algo` hash algorithm ('md5', 'sha1', 'sha256', or 'sha512')

### Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

### Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps value` Set the `allowed-impure-host-deps` setting.
- `--allowed-uris value` Set the `allowed-uris` setting.
- `--allowed-users value` Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt value` Set the `bash-prompt` setting.
- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.

- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.

- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.

- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.

- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.

- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.

- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.

- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix hash path` - print cryptographic hash of the NAR serialisation of a path

## Synopsis

`nix hash path [option...] paths...`

# Options

- `--base16` Print the hash in base-16 format.
- `--base32` Print the hash in base-32 (Nix-specific) format.
- `--base64` Print the hash in base-64 format.
- `--sri` Print the hash in SRI format.
- `--type hash-algo` hash algorithm ('md5', 'sha1', 'sha256', or 'sha512')

### Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

### Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps value` Set the `allowed-impure-host-deps` setting.
- `--allowed-uris value` Set the `allowed-uris` setting.
- `--allowed-users value` Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt value` Set the `bash-prompt` setting.
- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.

- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.

- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.

- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.

- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.

- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.

- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.

- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix hash to-base16` - convert a hash to base-16 representation

## Synopsis

`nix hash to-base16 [option...] strings...`

# Options

- `--type hash-algo` hash algorithm ('md5', 'sha1', 'sha256', or 'sha512'). Optional as can also be gotten from SRI hash itself.

### Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

### Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting *name* to *value* (overriding

`nix.conf`).

- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.

- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors value` Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls value` Append to the `ignored-acls` setting.
- `--extra-nix-path value` Append to the `nix-path` setting.
- `--extra-platforms value` Set the `extra-platforms` setting.
- `--extra-plugin-files value` Append to the `plugin-files` setting.
- `--extra-sandbox-paths value` Append to the `sandbox-paths` setting.

- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.

- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.

- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.

- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.

- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.

- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix hash to-base32` - convert a hash to base-32 representation

## Synopsis

`nix hash to-base32 [option...] strings...`

# Options

- `--type hash-algo` hash algorithm ('md5', 'sha1', 'sha256', or 'sha512'). Optional as can also be gotten from SRI hash itself.

### Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

### Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting *name* to *value* (overriding

`nix.conf`).

- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.

- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors value` Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls value` Append to the `ignored-acls` setting.
- `--extra-nix-path value` Append to the `nix-path` setting.
- `--extra-platforms value` Set the `extra-platforms` setting.
- `--extra-plugin-files value` Append to the `plugin-files` setting.
- `--extra-sandbox-paths value` Append to the `sandbox-paths` setting.

- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.

- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.

- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.

- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.

- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.

- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix hash to-base64` - convert a hash to base-64 representation

## Synopsis

`nix hash to-base64 [option...] strings...`

# Options

- `--type hash-algo` hash algorithm ('md5', 'sha1', 'sha256', or 'sha512'). Optional as can also be gotten from SRI hash itself.

### Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

### Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting *name* to *value* (overriding

`nix.conf`).

- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.

- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.

- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.

- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.

- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.

- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.

- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.

- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix hash to-sri` - convert a hash to SRI representation

# Synopsis

`nix hash to-sri [option...] strings...`

# Options

- `--type hash-algo` hash algorithm ('md5', 'sha1', 'sha256', or 'sha512'). Optional as can also be gotten from SRI hash itself.

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding

`nix.conf`).

- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.

- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors value` Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls value` Append to the `ignored-acls` setting.
- `--extra-nix-path value` Append to the `nix-path` setting.
- `--extra-platforms value` Set the `extra-platforms` setting.
- `--extra-plugin-files value` Append to the `plugin-files` setting.
- `--extra-sandbox-paths value` Append to the `sandbox-paths` setting.

- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.

- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.

- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.

- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.

- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.

- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

**Warning**

This program is **experimental** and its interface is subject to change.

---

# Name

`nix help` - show help about `nix` or a particular subcommand

# Synopsis

`nix help [option...] subcommand...`

# Examples

- Show help about `nix` in general:

```
nix help
```

- Show help about a particular subcommand:

```
nix help flake info
```

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix help-stores` - show help about store types and their settings

## Synopsis

`nix help-stores [option...]`

Nix supports different types of stores. These are described below.

## Store URL format

Stores are specified using a URL-like syntax. For example, the command

```
nix path-info --store https://cache.nixos.org/ --json \
/nix/store/a7gvj343m05j2s32xcnwr35v31ynlypr-coreutils-9.1
```

fetches information about a store path in the HTTP binary cache located at `https://cache.nixos.org/`, which is a type of store.

Store URLs can specify **store settings** using URL query strings, i.e. by appending `?name1=value1&name2=value2&...` to the URL. For instance,

```
--store ssh://machine.example.org?ssh-key=/path/to/my/key
```

tells Nix to access the store on a remote machine via the SSH protocol, using `/path/to/my/key` as the SSH private key. The supported settings for each store type are documented below.

The special store URL `auto` causes Nix to automatically select a store as follows:

- Use the **local store** `/nix/store` if `/nix/var/nix` is writable by the current user.
- Otherwise, if `/nix/var/nix/daemon-socket/socket` exists, **connect to the Nix daemon**

listening on that socket.

- Otherwise, on Linux only, use the **local chroot store** `~/.local/share/nix/root`, which will be created automatically if it does not exist.
- Otherwise, use the **local store** `/nix/store`.

## Dummy Store

**Store URL format:** `dummy://`

This store type represents a store that contains no store paths and cannot be written to. It's useful when you want to use the Nix evaluator when no actual Nix store exists, e.g.

```
nix eval --store dummy:// --expr '1 + 2'
```

**Settings:**

- **path-info-cache-size**

Size of the in-memory store path metadata cache.

**Default:** `65536`

- **priority**

Priority of this store when used as a substituter. A lower value means a higher priority.

**Default:** `0`

- **store**

Logical location of the Nix store, usually `/nix/store`. Note that you can only copy store paths between stores if they have the same **store** setting.

**Default:** `/nix/store`

- **system-features**

Optional features that the system this store builds on implements (like "kvm").

**Default:** `benchmark big-parallel kvm nixos-test uid-range`

- **trusted**

Whether paths from this store can be used as substitutes even if they are not signed by a key listed in the `trusted-public-keys` setting.

**Default:** `false`

- `want-mass-query`

Whether this store (when used as a substituter) can be queried efficiently for path validity.

**Default:** `false`

## Experimental SSH Store

**Store URL format:** `ssh-ng://[username@]hostname`

Experimental store type that allows full access to a Nix store on a remote machine.

**Settings:**

- `base64-ssh-public-host-key`

The public host key of the remote machine.

**Default:** `empty`

- `compress`

Whether to enable SSH compression.

**Default:** `false`

- `max-connection-age`

Maximum age of a connection before it is closed.

**Default:** `4294967295`

- `max-connections`

Maximum number of concurrent connections to the Nix daemon.

**Default:** `1`

- `path-info-cache-size`

Size of the in-memory store path metadata cache.

**Default:** `65536`

- `priority`

Priority of this store when used as a substituter. A lower value means a higher priority.

**Default:** `0`

- `remote-program`

Path to the `nix-daemon` executable on the remote machine.

**Default:** `nix-daemon`

- `remote-store`

`Store URL` to be used on the remote machine. The default is `auto` (i.e. use the Nix daemon or `/nix/store` directly).

**Default:** `empty`

- `ssh-key`

Path to the SSH private key used to authenticate to the remote machine.

**Default:** `empty`

- `store`

Logical location of the Nix store, usually `/nix/store`. Note that you can only copy store paths between stores if they have the same `store` setting.

**Default:** `/nix/store`

- `system-features`

Optional features that the system this store builds on implements (like "kvm").

**Default:** `benchmark big-parallel kvm nixos-test uid-range`

- `trusted`

Whether paths from this store can be used as substitutes even if they are not signed by a key listed in the `trusted-public-keys` setting.

**Default:** `false`

- `want-mass-query`

Whether this store (when used as a substituter) can be queried efficiently for path validity.

**Default:** `false`

## HTTP Binary Cache Store

**Store URL format:** `http://...`, `https://...`

This store allows a binary cache to be accessed via the HTTP protocol.

**Settings:**

- `compression`

NAR compression method (`xz`, `bzip2`, `gzip`, `zstd`, or `none`).

**Default:** `xz`

- `compression-level`

The *preset level* to be used when compressing NARs. The meaning and accepted values depend on the compression method selected. `-1` specifies that the default compression level should be used.

**Default:** `-1`

- `index-debug-info`

Whether to index DWARF debug info files by build ID. This allows `dwarf` to fetch debug info on demand

**Default:** `false`

- `local-nar-cache`

Path to a local cache of NARs fetched from this binary cache, used by commands such as `nix store cat`.

**Default:** `empty`

- **parallel-compression**

Enable multi-threaded compression of NARs. This is currently only available for `xz` and `zstd`.

**Default:** `false`

- **path-info-cache-size**

Size of the in-memory store path metadata cache.

**Default:** `65536`

- **priority**

Priority of this store when used as a substituter. A lower value means a higher priority.

**Default:** `0`

- **secret-key**

Path to the secret key used to sign the binary cache.

**Default:** `empty`

- **store**

Logical location of the Nix store, usually `/nix/store`. Note that you can only copy store paths between stores if they have the same `store` setting.

**Default:** `/nix/store`

- **system-features**

Optional features that the system this store builds on implements (like "kvm").

**Default:** `benchmark big-parallel kvm nixos-test uid-range`

- **trusted**

Whether paths from this store can be used as substitutes even if they are not signed by a key listed in the `trusted-public-keys` setting.

**Default:** `false`

- **want-mass-query**

Whether this store (when used as a substituter) can be queried efficiently for path validity.

**Default:** `false`

- `write-nar-listing`

Whether to write a JSON file that lists the files in each NAR.

**Default:** `false`

## Local Binary Cache Store

**Store URL format:** `file:// path`

This store allows reading and writing a binary cache stored in *path* in the local filesystem. If *path* does not exist, it will be created.

For example, the following builds or downloads `nixpkgs#hello` into the local store and then copies it to the binary cache in `/tmp/binary-cache`:

```
nix copy --to file:///tmp/binary-cache nixpkgs#hello
```

**Settings:**

- `compression`

NAR compression method (`xz`, `bzip2`, `gzip`, `zstd`, or `none`).

**Default:** `xz`

- `compression-level`

The *preset level* to be used when compressing NARs. The meaning and accepted values depend on the compression method selected. `-1` specifies that the default compression level should be used.

**Default:** `-1`

- `index-debug-info`

Whether to index DWARF debug info files by build ID. This allows `dwarf` to fetch debug info on demand

**Default:** `false`

- **local-nar-cache**

Path to a local cache of NARs fetched from this binary cache, used by commands such as `nix store cat`.

**Default:** `empty`

- **parallel-compression**

Enable multi-threaded compression of NARs. This is currently only available for `xz` and `zstd`.

**Default:** `false`

- **path-info-cache-size**

Size of the in-memory store path metadata cache.

**Default:** `65536`

- **priority**

Priority of this store when used as a substituter. A lower value means a higher priority.

**Default:** `0`

- **secret-key**

Path to the secret key used to sign the binary cache.

**Default:** `empty`

- **store**

Logical location of the Nix store, usually `/nix/store`. Note that you can only copy store paths between stores if they have the same `store` setting.

**Default:** `/nix/store`

- **system-features**

Optional features that the system this store builds on implements (like "kvm").

**Default:** `benchmark big-parallel kvm nixos-test uid-range`

- **trusted**

Whether paths from this store can be used as substitutes even if they are not signed by a key listed in the `trusted-public-keys` setting.

**Default:** `false`

- `want-mass-query`

Whether this store (when used as a substituter) can be queried efficiently for path validity.

**Default:** `false`

- `write-nar-listing`

Whether to write a JSON file that lists the files in each NAR.

**Default:** `false`

## Local Daemon Store

**Store URL format:** `daemon`, `unix:// path`

This store type accesses a Nix store by talking to a Nix daemon listening on the Unix domain socket *path*. The store pseudo-URL `daemon` is equivalent to `unix:///nix/var/nix/daemon-socket/socket`.

**Settings:**

- `log`

directory where Nix will store log files.

**Default:** `/nix/var/log/nix`

- `max-connection-age`

Maximum age of a connection before it is closed.

**Default:** `4294967295`

- `max-connections`

Maximum number of concurrent connections to the Nix daemon.

**Default:** `1`

- **path-info-cache-size**

Size of the in-memory store path metadata cache.

**Default:** 65536

- **priority**

Priority of this store when used as a substituter. A lower value means a higher priority.

**Default:** 0

- **real**

Physical path of the Nix store.

**Default:** /nix/store

- **root**

Directory prefixed to all other paths.

**Default:** ``

- **state**

Directory where Nix will store state.

**Default:** /dummy

- **store**

Logical location of the Nix store, usually /nix/store. Note that you can only copy store paths between stores if they have the same **store** setting.

**Default:** /nix/store

- **system-features**

Optional features that the system this store builds on implements (like "kvm").

**Default:** benchmark big-parallel kvm nixos-test uid-range

- **trusted**

Whether paths from this store can be used as substitutes even if they are not signed by a key listed in the **trusted-public-keys** setting.

**Default:** `false`

- `want-mass-query`

Whether this store (when used as a substituter) can be queried efficiently for path validity.

**Default:** `false`

## Local Store

**Store URL format:** `local`, `root`

This store type accesses a Nix store in the local filesystem directly (i.e. not via the Nix daemon). `root` is an absolute path that is prefixed to other directories such as the Nix store directory. The store pseudo-URL `local` denotes a store that uses `/` as its root directory.

A store that uses a `root` other than `/` is called a *chroot store*. With such stores, the store directory is "logically" still `/nix/store`, so programs stored in them can only be built and executed by `chroot`-ing into `root`. Chroot stores only support building and running on Linux when `mount namespaces` and `user namespaces` are enabled.

For example, the following uses `/tmp/root` as the chroot environment to build or download `nixpkgs#hello` and then execute it:

```
nix run --store /tmp/root nixpkgs#hello
Hello, world!
```

Here, the "physical" store location is `/tmp/root/nix/store`, and Nix's store metadata is in `/tmp/root/nix/var/nix/db`.

It is also possible, but not recommended, to change the "logical" location of the Nix store from its default of `/nix/store`. This makes it impossible to use default substituters such as <https://cache.nixos.org/>, and thus you may have to build everything locally. Here is an example:

```
nix build --store 'local?store=/tmp/my-nix/store&state=/tmp/my-nix/state&
log=/tmp/my-nix/log' nixpkgs#hello
```

**Settings:**

- `log`

directory where Nix will store log files.

**Default:** `/nix/var/log/nix`

- `path-info-cache-size`

Size of the in-memory store path metadata cache.

**Default:** `65536`

- `priority`

Priority of this store when used as a substituter. A lower value means a higher priority.

**Default:** `0`

- `read-only`

Allow this store to be opened when its `database` is on a read-only filesystem.

Normally Nix will attempt to open the store database in read-write mode, even for querying (when write access is not needed), causing it to fail if the database is on a read-only filesystem.

Enable read-only mode to disable locking and open the SQLite database with the `immutable` parameter set.

---

**Warning** Do not use this unless the filesystem is read-only.

Using it when the filesystem is writable can cause incorrect query results or corruption errors if the database is changed by another process. While the filesystem the database resides on might appear to be read-only, consider whether another user or system might have write access to it.

---

**Default:** `false`

- `real`

Physical path of the Nix store.

**Default:** `/nix/store`

- `require-sigs`

Whether store paths copied into this store should have a trusted signature.

**Default:** `true`

- `root`

Directory prefixed to all other paths.

**Default:** ````

- `state`

Directory where Nix will store state.

**Default:** `/dummy`

- `store`

Logical location of the Nix store, usually `/nix/store`. Note that you can only copy store paths between stores if they have the same `store` setting.

**Default:** `/nix/store`

- `system-features`

Optional features that the system this store builds on implements (like "kvm").

**Default:** `benchmark big-parallel kvm nixos-test uid-range`

- `trusted`

Whether paths from this store can be used as substitutes even if they are not signed by a key listed in the `trusted-public-keys` setting.

**Default:** `false`

- `want-mass-query`

Whether this store (when used as a substituter) can be queried efficiently for path validity.

**Default:** `false`

## S3 Binary Cache Store

**Store URL format:** `s3:// bucket-name`

This store allows reading and writing a binary cache stored in an AWS S3 bucket.

## Settings:

- `buffer-size`

Size (in bytes) of each part in multi-part uploads.

**Default:** `5242880`

- `compression`

NAR compression method (`xz`, `bzip2`, `gzip`, `zstd`, or `none`).

**Default:** `xz`

- `compression-level`

The *preset level* to be used when compressing NARs. The meaning and accepted values depend on the compression method selected. `-1` specifies that the default compression level should be used.

**Default:** `-1`

- `endpoint`

The URL of the endpoint of an S3-compatible service such as MinIO. Do not specify this setting if you're using Amazon S3.

---

### Note

This endpoint must support HTTPS and will use path-based addressing instead of virtual host based addressing.

---

**Default:** `empty`

- `index-debug-info`

Whether to index DWARF debug info files by build ID. This allows `dwarf` to fetch debug info on demand

**Default:** `false`

- `local-nar-cache`

Path to a local cache of NARs fetched from this binary cache, used by commands such as `nix store cat`.

**Default:** `empty`

- `log-compression`

Compression method for `log/*` files. It is recommended to use a compression method supported by most web browsers (e.g. `brotli`).

**Default:** `empty`

- `ls-compression`

Compression method for `.ls` files.

**Default:** `empty`

- `multipart-upload`

Whether to use multi-part uploads.

**Default:** `false`

- `narinfo-compression`

Compression method for `.narinfo` files.

**Default:** `empty`

- `parallel-compression`

Enable multi-threaded compression of NARs. This is currently only available for `xz` and `zstd`.

**Default:** `false`

- `path-info-cache-size`

Size of the in-memory store path metadata cache.

**Default:** `65536`

- `priority`

Priority of this store when used as a substituter. A lower value means a higher priority.

**Default:** 0

- `profile`

The name of the AWS configuration profile to use. By default Nix will use the `default` profile.

**Default:** *empty*

- `region`

The region of the S3 bucket. If your bucket is not in `us-east-1`, you should always explicitly specify the region parameter.

**Default:** `us-east-1`

- `scheme`

The scheme used for S3 requests, `https` (default) or `http`. This option allows you to disable HTTPS for binary caches which don't support it.

---

**Note**

HTTPS should be used if the cache might contain sensitive information.

---

**Default:** *empty*

- `secret-key`

Path to the secret key used to sign the binary cache.

**Default:** *empty*

- `store`

Logical location of the Nix store, usually `/nix/store`. Note that you can only copy store paths between stores if they have the same `store` setting.

**Default:** `/nix/store`

- `system-features`

Optional features that the system this store builds on implements (like "kvm").

**Default:** `benchmark big-parallel kvm nixos-test uid-range`

- **trusted**

Whether paths from this store can be used as substitutes even if they are not signed by a key listed in the `trusted-public-keys` setting.

**Default:** `false`

- **want-mass-query**

Whether this store (when used as a substituter) can be queried efficiently for path validity.

**Default:** `false`

- **write-nar-listing**

Whether to write a JSON file that lists the files in each NAR.

**Default:** `false`

## SSH Store

**Store URL format:** `ssh://[username@]hostname`

This store type allows limited access to a remote store on another machine via SSH.

### Settings:

- **base64-ssh-public-host-key**

The public host key of the remote machine.

**Default:** `empty`

- **compress**

Whether to enable SSH compression.

**Default:** `false`

- **max-connections**

Maximum number of concurrent SSH connections.

**Default:** `1`

- **path-info-cache-size**

Size of the in-memory store path metadata cache.

**Default:** 65536

- **priority**

Priority of this store when used as a substituter. A lower value means a higher priority.

**Default:** 0

- **remote-program**

Path to the `nix-store` executable on the remote machine.

**Default:** `nix-store`

- **remote-store**

**Store URL** to be used on the remote machine. The default is `auto` (i.e. use the Nix daemon or `/nix/store` directly).

**Default:** `empty`

- **ssh-key**

Path to the SSH private key used to authenticate to the remote machine.

**Default:** `empty`

- **store**

Logical location of the Nix store, usually `/nix/store`. Note that you can only copy store paths between stores if they have the same `store` setting.

**Default:** `/nix/store`

- **system-features**

Optional features that the system this store builds on implements (like "kvm").

**Default:** `benchmark big-parallel kvm nixos-test uid-range`

- **trusted**

Whether paths from this store can be used as substitutes even if they are not signed by

a key listed in the `trusted-public-keys` setting.

**Default:** `false`

- `want-mass-query`

Whether this store (when used as a substituter) can be queried efficiently for path validity.

**Default:** `false`

---

**Warning**

This program is **experimental** and its interface is subject to change.

---

# Name

`nix key` - generate and convert Nix signing keys

# Synopsis

`nix key [option...] subcommand`

where *subcommand* is one of the following:

- `nix key convert-secret-to-public` - generate a public key for verifying store paths from a secret key read from standard input
- `nix key generate-secret` - generate a secret key for signing store paths

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix key convert-secret-to-public` - generate a public key for verifying store paths from a secret key read from standard input

# Synopsis

`nix key convert-secret-to-public [option...]`

# Examples

- Convert a secret key to a public key:

```
echo cache.example.org-
0:E7lAO+MsPwTffPXsdPtW8GKui/5ho4KQHVcAGnX+Tti1V4dUxoVoqLyWJ4YESuZJwQ67GVIks
Dt47og+tPVUZw== \
| nix key convert-secret-to-public
cache.example.org-0:tVeHVMaFaKi8lieGBErmScEOuxlSJLA7e06IPrT1VGc=
```

# Description

This command reads a Ed25519 secret key from standard input, and writes the corresponding public key to standard output. For more details, see [nix key generate-secret](#).

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix key generate-secret` - generate a secret key for signing store paths

# Synopsis

`nix key generate-secret [option...]`

# Examples

- Generate a new secret key:

```
nix key generate-secret --key-name cache.example.org-1 > ./secret-key
```

We can then use this key to sign the closure of the Hello package:

```
nix build nixpkgs#hello
nix store sign --key-file ./secret-key --recursive ./result
```

Finally, we can verify the store paths using the corresponding public key:

```
nix store verify --trusted-public-keys $(nix key convert-secret-to-public
< ./secret-key) ./result
```

# Description

This command generates a new Ed25519 secret key for signing store paths and prints it on standard output. Use `nix key convert-secret-to-public` to get the corresponding public key for verifying signed store paths.

The mandatory argument `--key-name` specifies a key name (such as `cache.example.org-1`). It is used to look up keys on the client when it verifies signatures. It can be anything, but it's suggested to use the host name of your cache (e.g. `cache.example.org`) with a suffix denoting the number of the key (to be incremented every time you need to revoke a key).

## Format

Both secret and public keys are represented as the key name followed by a base-64 encoding of the Ed25519 key data, e.g.

```
cache.example.org-
0:E7lA0+MsPwTFFPXsdPtW8GKui/5ho4KQHVcAGnX+Tti1V4dUxoVoqLyWJ4YESuZJwQ67GVIksDt47
og+tPVUZw==
```

## Options

- `--key-name` *name* Identifier of the key (e.g. `cache.example.org-1`).

### Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

### Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding

`nix.conf`).

- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.

- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors value` Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls value` Append to the `ignored-acls` setting.
- `--extra-nix-path value` Append to the `nix-path` setting.
- `--extra-platforms value` Set the `extra-platforms` setting.
- `--extra-plugin-files value` Append to the `plugin-files` setting.
- `--extra-sandbox-paths value` Append to the `sandbox-paths` setting.

- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.

- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.

- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.

- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.

- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.

- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix log` - show the build log of the specified packages or paths, if available

## Synopsis

`nix log [option...] installable`

# Examples

- Get the build log of GNU Hello:

```
nix log nixpkgs#hello
```

- Get the build log of a specific store path:

```
nix log /nix/store/lmngj4wcm9rkv3w4dfhzhcij3195hiq-thunderbird-52.2.1
```

- Get a build log from a specific binary cache:

```
nix log --store https://cache.nixos.org nixpkgs#hello
```

# Description

This command prints the log of a previous build of the *installable* on standard output.

Nix looks for build logs in two places:

- In the directory `/nix/var/log/nix/drvs`, which contains logs for locally built

derivations.

- In the binary caches listed in the `substituters` setting. Logs should be named `<cache>/log/<base-name-of-store-path>`, where `store-path` is a derivation, e.g. <https://cache.nixos.org/log/dvmig8jgrdapvbyxb1rprckdmdqx08kv-hello-2.10drv>. For non-derivation store paths, Nix will first try to determine the deriver by fetching the `.narinfo` file for this store path.

# Options

## Common evaluation options:

- `--arg` *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr` *name string* Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store` *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include` / `-I` *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for nix-channel](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding

`nix.conf`).

- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options that change the interpretation of `installables`:

- `--expr` *expr* Interpret `installables` as attribute paths relative to the Nix expression *expr*.
- `--file` / `-f` *file* Interpret `installables` as attribute paths relative to the Nix expression stored in *file*. If *file* is the character -, then a Nix expression will be read from standard input. Implies `--impure`.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.

- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.
- `--build-hook value` Set the `build-hook` setting.
- `--build-poll-interval value` Set the `build-poll-interval` setting.
- `--build-users-group value` Set the `build-users-group` setting.
- `--builders value` Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.

- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.

- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.

- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.

- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-`

groups setting.

- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.

- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix nar` - create or inspect NAR files

# Synopsis

`nix nar [option...] subcommand`

where *subcommand* is one of the following:

- `nix nar cat` - print the contents of a file inside a NAR file on stdout
- `nix nar dump-path` - serialise a path to stdout in NAR format
- `nix nar ls` - show information about a path inside a NAR file

# Description

`nix nar` provides several subcommands for creating and inspecting *Nix Archives* (NARs).

# File format

For the definition of the NAR file format, see Figure 5.2 in <https://edolstra.github.io/pubs/phd-thesis.pdf>.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix nar cat` - print the contents of a file inside a NAR file on stdout

# Synopsis

`nix nar cat [option...] nar path`

# Examples

- List a file in a NAR and pipe it through `gunzip`:

```
nix nar cat ./hello.nar /share/man/man1/hello.1.gz | gunzip
.\\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.46.4.
.TH HELLO "1" "November 2014" "hello 2.10" "User Commands"
...
```

# Description

This command prints on standard output the contents of the regular file *path* inside the NAR file *nar*.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix nar dump-path` - serialise a path to stdout in NAR format

# Synopsis

`nix nar dump-path [option...] path`

# Examples

- To serialise directory `foo` as a NAR:

```
nix nar dump-path ./foo > foo.nar
```

# Description

This command generates a NAR file containing the serialisation of *path*, which must contain only regular files, directories and symbolic links. The NAR is written to standard output.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix nar ls` - show information about a path inside a NAR file

# Synopsis

`nix nar ls [option...] nar path`

# Examples

- To list a specific file in a NAR:

```
nix nar ls --long ./hello.nar /bin/hello
-r-xr-xr-x 38184 hello
```

- To recursively list the contents of a directory inside a NAR, in JSON format:

```
nix nar ls --json --recursive ./hello.nar /bin
{"type": "directory", "entries": {"hello": {
 "type": "regular", "size": 38184, "executable": true, "narOffset": 400}}}
```

# Description

This command shows information about a *path* inside NAR file *nar*.

# Options

- `--directory` / `-d` Show directories rather than their contents.
- `--json` Produce output in JSON format, suitable for consumption by another program.
- `--long` / `-l` Show detailed file information.
- `--recursive` / `-R` List subdirectories recursively.

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.

- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.

- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.

- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.
- `--max-silent-time` *value* Set the `max-silent-time` setting.
- `--max-substitution-jobs` *value* Set the `max-substitution-jobs` setting.
- `--min-free` *value* Set the `min-free` setting.
- `--min-free-check-interval` *value* Set the `min-free-check-interval` setting.

- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.

- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.

- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.
- `--start-id value` Set the `start-id` setting.

- `--store value` Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix path-info` - query information about store paths

## Synopsis

`nix path-info [option...] installables...`

## Examples

- Print the store path produced by `nixpkgs#hello`:

```
nix path-info nixpkgs#hello
/nix/store/v5sv61sszx301i0x6xysaqzla09nksnd-hello-2.10
```

- Show the closure sizes of every path in the current NixOS system closure, sorted by size:

```
nix path-info --recursive --closure-size /run/current-system | sort -nk2
/nix/store/hl5xwp9kdrd1zkm0idm3kkby9q66z404-empty
96
/nix/store/27324qvqhnxj3rncazmxc4mwy79kz8ha-nameservers
112
...
/nix/store/539jkw9a8dyry7clcv60gk6na816j7y8-etc
5783255504
/nix/store/zqamz3cz4dbzfihki2mk7a63mbkxz9xq-nixos-system-machine-
20.09.20201112.3090c65 5887562256
```

- Show a package's closure size and all its dependencies with human readable sizes:

```
nix path-info --recursive --size --closure-size --human-readable
nixpkgs#rustc
/nix/store/01rrgsg5zk3cds0xgdsq40zpk6g51dz9-ncurses-6.2-dev 386.7K
69.1M
/nix/store/0q783wnvixpqz6dxjp16nw296avgczam-libpfm-4.11.0 5.9M
37.4M
...

```

- Check the existence of a path in a binary cache:

```
nix path-info --recursive /nix/store/blzxgyvrk32ki6xga10phr4sby2xf25q-
geeqie-1.5.1 --store https://cache.nixos.org/
path '/nix/store/blzxgyvrk32ki6xga10phr4sby2xf25q-geeqie-1.5.1' is not
valid
```

- Print the 10 most recently added paths (using --json and the jq(1) command):

```
nix path-info --json --all | jq -r 'sort_by(.registrationTime)[-11:-
1] [] .path'
```

- Show the size of the entire Nix store:

```
nix path-info --json --all | jq 'map(.narSize) | add'
49812020936
```

- Show every path whose closure is bigger than 1 GB, sorted by closure size:

```
nix path-info --json --all --closure-size \
| jq 'map(select(.closureSize > 1e9)) | sort_by(.closureSize) |
map([.path, .closureSize])'
[
 ...,
 [
 "/nix/store/zqamz3cz4dbzfihki2mk7a63mbkxz9xq-nixos-system-machine-
20.09.20201112.3090c65",
 5887562256
]
]
```

- Print the path of the [store derivation](#) produced by `nixpkgs#hello`:

```
nix path-info --derivation nixpkgs#hello
/nix/store/s6rn4jz1sin56rf4qj5b5v8jxjm32hlk-hello-2.10.drv
```

# Description

This command shows information about the store paths produced by [installables](#), or about all paths in the store if you pass `--all`.

By default, this command only prints the store paths. You can get additional information by passing flags such as `--closure-size`, `--size`, `--sigs` or `--json`.

---

## Warning

Note that `nix path-info` does not build or substitute the *installables* you specify.

Thus, if the corresponding store paths don't already exist, this command will fail. You can use `nix build` to ensure that they exist.

---

# Options

- `--closure-size` / `-S` Print the sum of the sizes of the NAR serialisations of the closure of each path.
- `--human-readable` / `-h` With `-s` and `-S`, print sizes in a human-friendly format such as `5.67G`.
- `--json` Produce output in JSON format, suitable for consumption by another program.
- `--sigs` Show signatures.
- `--size` / `-s` Print the size of the NAR serialisation of each path.
- `--stdin` Read installables from the standard input. No default installable applied.

## Common evaluation options:

- `--arg` *name expr* Pass the value *expr* as the argument *name* to Nix functions.

- **--argstr** *name string* Pass the string *string* as the argument *name* to Nix functions.
- **--debugger** Start an interactive environment if evaluation fails.
- **--eval-store** *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- **--impure** Allow access to mutable paths and repositories.
- **--include** / **-I** *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for](#)

`nix-channel`) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.

- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

### Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

### Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

### Options that change the interpretation of `installables`:

- `--all` Apply the operation to every store path.
- `--derivation` Operate on the `store derivation` rather than its outputs.
- `--expr expr` Interpret `installables` as attribute paths relative to the Nix expression `expr`.
- `--file / -f file` Interpret `installables` as attribute paths relative to the Nix expression stored in `file`. If `file` is the character `-`, then a Nix expression will be read from standard

input. Implies `--impure`.

- `--recursive` / `-r` Apply operation to closure of the specified paths.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.

- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors value` Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls value` Append to the `ignored-acls` setting.
- `--extra-nix-path value` Append to the `nix-path` setting.
- `--extra-platforms value` Set the `extra-platforms` setting.
- `--extra-plugin-files value` Append to the `plugin-files` setting.
- `--extra-sandbox-paths value` Append to the `sandbox-paths` setting.
- `--extra-secret-key-files value` Append to the `secret-key-files` setting.

- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.

- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.

- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.

- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.

- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.

- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

**nix print-dev-env** - print shell code that can be sourced by bash to reproduce the build environment of a derivation

# Synopsis

**nix print-dev-env** [*option...*] *installable*

# Examples

- Apply the build environment of GNU hello to the current shell:

```
. <(nix print-dev-env nixpkgs#hello)
```

- Get the build environment in JSON format:

```
nix print-dev-env nixpkgs#hello --json
```

The output will look like this:

```
{
 "bashFunctions": {
 "buildPhase": "# \n runHook preBuild;\n...",
 ...
 },
 "variables": {
 "src": {
 "type": "exported",
 "value": "/nix/store/3x7dwzq014bblazs7kq20p9hyzz0qh8g-hello-
2.10.tar.gz"
 },
 "postUnpackHooks": {
 "type": "array",
 "value": ["_updateSourceDateEpochFromSourceRoot"]
 },
 ...
 }
}
```

# Description

This command prints a shell script that can be sourced by `bash` and that sets the variables and shell functions defined by the build process of `installable`. This allows you to get a similar build environment in your current shell rather than in a subshell (as with `nix develop`).

With `--json`, the output is a JSON serialisation of the variables and functions defined by the build process.

# Options

- `--json` Produce output in JSON format, suitable for consumption by another program.
- `--profile path` The profile to operate on.
- `--redirect installable outputs-dir` Redirect a store path to a mutable location.

**Common evaluation options:**

- **--arg** *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- **--argstr** *name string* Pass the string *string* as the argument *name* to Nix functions.
- **--debugger** Start an interactive environment if evaluation fails.
- **--eval-store** *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- **--impure** Allow access to mutable paths and repositories.
- **--include** / **-I** *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using **paths** enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.

- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options that change the interpretation of `installables`:

- `--expr expr` Interpret `installables` as attribute paths relative to the Nix expression `expr`.
- `--file / -f file` Interpret `installables` as attribute paths relative to the Nix expression stored in `file`. If `file` is the character `-`, then a Nix expression will be read from standard input. Implies `--impure`.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.

- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors value` Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls value` Append to the `ignored-acls` setting.
- `--extra-nix-path value` Append to the `nix-path` setting.
- `--extra-platforms value` Set the `extra-platforms` setting.
- `--extra-plugin-files value` Append to the `plugin-files` setting.
- `--extra-sandbox-paths value` Append to the `sandbox-paths` setting.
- `--extra-secret-key-files value` Append to the `secret-key-files` setting.
- `--extra-substituters value` Append to the `substituters` setting.
- `--extra-system-features value` Append to the `system-features` setting.
- `--extra-trusted-public-keys value` Append to the `trusted-public-keys` setting.

- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.

- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.

- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.

- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.

- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.
- `--start-id value` Set the `start-id` setting.
- `--store value` Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix profile` - manage Nix profiles

# Synopsis

`nix profile [option...] subcommand`

where *subcommand* is one of the following:

- `nix profile diff-closures` - show the closure difference between each version of a profile
- `nix profile history` - show all versions of a profile
- `nix profile install` - install a package into a profile
- `nix profile list` - list installed packages
- `nix profile remove` - remove packages from a profile
- `nix profile rollback` - roll back to the previous version or a specified version of a profile
- `nix profile upgrade` - upgrade packages using their most recent flake
- `nix profile wipe-history` - delete non-current versions of a profile

# Description

`nix profile` allows you to create and manage *Nix profiles*. A Nix profile is a set of packages that can be installed and upgraded independently from each other. Nix profiles are versioned, allowing them to be rolled back easily.

# Files

# 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
/profile -> profile-7-link
lrwxrwxrwx 1 alice users 51 Oct 28 16:18 /home/alice/.local/state/nix/profiles
/profile-5-link -> /nix/store/q69xad13ghpf7ir87h0b2gd28lafjj1j-profile
lrwxrwxrwx 1 alice users 51 Oct 29 13:20 /home/alice/.local/state/nix/profiles
/profile-6-link -> /nix/store/6bvhpysd7vwz7k3b0pndn7ifi5xr32dg-profile
lrwxrwxrwx 1 alice users 51 Nov 25 14:35 /home/alice/.local/state/nix/profiles
/profile-7-link -> /nix/store/mp0x6xnsg0b8qhswy6riqvimai4gm677-profile
```

Each of these symlinks is a root for the Nix garbage collector.

The contents of the store path corresponding to each version of the profile is a tree of symlinks to the files of the installed packages, e.g.

```
$ ll -R ~eelco/.local/state/nix/profiles/profile-7-link/
/home/eelco/.local/state/nix/profiles/profile-7-link/:
total 20
dr-xr-xr-x 2 root root 4096 Jan 1 1970 bin
-r--r--r-- 2 root root 1402 Jan 1 1970 manifest.nix
dr-xr-xr-x 4 root root 4096 Jan 1 1970 share

/home/eelco/.local/state/nix/profiles/profile-7-link/bin:
total 20
lwxrwxrwx 5 root root 79 Jan 1 1970 chromium -> /nix/store
/ijm5k0zqisvkdwjkc77mb9qzb35xfi4m-chromium-86.0.4240.111/bin/chromium
lwxrwxrwx 7 root root 87 Jan 1 1970 spotify -> /nix/store
/w9182874m1bl56smpr3m5zjj36jhp3rn-spotify-1.1.26.501.gbe1le53b-15/bin/spotify
lwxrwxrwx 3 root root 79 Jan 1 1970 zoom-us -> /nix/store
/wbhg2ga8f3h87s9h5k0slxk0m81m4cxl-zoom-us-5.3.469451.0927/bin/zoom-us

/home/eelco/.local/state/nix/profiles/profile-7-link/share/applications:
total 12
lwxrwxrwx 4 root root 120 Jan 1 1970 chromium-browser.desktop -> /nix/store
/4cf803y4vzfm3gyk3vzhzb2327v0kl8a-chromium-unwrapped-86.0.4240.111/share
/applications/chromium-browser.desktop
lwxrwxrwx 7 root root 110 Jan 1 1970 spotify.desktop -> /nix/store
/w9182874m1bl56smpr3m5zjj36jhp3rn-spotify-1.1.26.501.gbe1le53b-15/share
/applications/spotify.desktop
lwxrwxrwx 3 root root 107 Jan 1 1970 us.zoom.Zoom.desktop -> /nix/store
/wbhg2ga8f3h87s9h5k0slxk0m81m4cxl-zoom-us-5.3.469451.0927/share/applications
/us.zoom.Zoom.desktop

...
```

Each profile version contains a manifest file:

- `manifest.nix` used by `nix-env`.
- `manifest.json` used by `nix profile` (experimental).

## User profile link

A symbolic link to the user's current profile:

- `~/.nix-profile`
- `$XDG_STATE_HOME/nix/profile` if `use-xdg-base-directories` is set to `true`.

By default, this symlink points to:

- `$XDG_STATE_HOME/nix/profiles/profile` for regular users
- `$NIX_STATE_DIR/profiles/per-user/root/profile` for `root`

The `PATH` environment variable should include `/bin` subdirectory of the profile link (e.g.

`~/.nix-profile/bin`) for the user environment to be visible to the user. The [installer](#) sets this up by default, unless you enable [`use-xdg-base-directories`](#).

## Profile compatibility

---

### Warning

Once you have used [`nix profile`](#) you can no longer use [`nix-env`](#) without first deleting `$XDG_STATE_HOME/nix/profiles/profile`

---

Once you installed a package with [`nix profile`](#), you get the following error message when using [`nix-env`](#):

```
$ nix-env -f '<nixpkgs>' -iA 'hello'
error: nix-env
profile '/home/alice/.local/state/nix/profiles/profile' is incompatible with
'nix-env'; please use 'nix profile' instead
```

To migrate back to [`nix-env`](#) you can delete your current profile:

---

### Warning

This will delete packages that have been installed before, so you may want to back up this information before running the command.

---

```
$ rm -rf "${XDG_STATE_HOME-$HOME/.local/state}/nix/profiles/profile"
```

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix profile diff-closures` - show the closure difference between each version of a profile

# Synopsis

`nix profile diff-closures [option...]`

# Examples

- Show what changed between each version of the NixOS system profile:

```
nix profile diff-closures --profile /nix/var/nix/profiles/system
Version 13 -> 14:
 acpi-call: 2020-04-07-5.8.13 → 2020-04-07-5.8.14
 aws-sdk-cpp: -6723.1 KiB
 ...
Version 14 -> 15:
 acpi-call: 2020-04-07-5.8.14 → 2020-04-07-5.8.16
 attica: -996.2 KiB
 breeze-icons: -78713.5 KiB
 brotli: 1.0.7 → 1.0.9, +44.2 KiB
```

# Description

This command shows the difference between the closures of subsequent versions of a profile. See `nix store diff-closures` for details.

# Options

- `--profile path` The profile to operate on.

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.

- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps value` Set the `allowed-impure-host-deps` setting.
- `--allowed-uris value` Set the `allowed-uris` setting.
- `--allowed-users value` Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt value` Set the `bash-prompt` setting.
- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.
- `--build-hook value` Set the `build-hook` setting.
- `--build-poll-interval value` Set the `build-poll-interval` setting.
- `--build-users-group value` Set the `build-users-group` setting.
- `--builders value` Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.

- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.

- `--gc-reserved-space value` Set the `gc-reserved-space` setting.
- `--hashed-mirrors value` Set the `hashed-mirrors` setting.
- `--http-connections value` Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count value` Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls value` Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.

- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.

- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.

- `--plugin-files` *value* Set the `plugin-files` setting.
- `--post-build-hook` *value* Set the `post-build-hook` setting.
- `--pre-build-hook` *value* Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.

- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix profile history` - show all versions of a profile

# Synopsis

`nix profile history [option...]`

# Examples

- Show the changes between each version of your default profile:

```
nix profile history
Version 508 (2020-04-10):
 flake:nixpkgs#legacyPackages.x86_64-linux.awscli: ø -> 1.17.13

Version 509 (2020-05-16) <- 508:
 flake:nixpkgs#legacyPackages.x86_64-linux.awscli: 1.17.13 -> 1.18.211
```

# Description

This command shows what packages were added, removed or upgraded between subsequent versions of a profile. It only shows top-level packages, not dependencies; for that, use `nix profile diff-closures`.

The addition of a package to a profile is denoted by the string `ø -> version`, whereas the removal is denoted by `version -> ø`.

# Options

- `--profile path` The profile to operate on.

## Common evaluation options:

- `--arg name expr` Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr name string` Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store store-url` The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include / -I path` Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location.

The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- **--help** Show usage information.
- **--offline** Disable substituters and consider all previously downloaded files up-to-date.
- **--option** *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- **--refresh** Consider all previously downloaded files out-of-date.
- **--repair** During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- **--version** Show version information.

## Options to override configuration settings:

- **--accept-flake-config** Enable the `accept-flake-config` setting.
- **--access-tokens** *value* Set the `access-tokens` setting.
- **--allow-dirty** Enable the `allow-dirty` setting.
- **--allow-import-from-derivation** Enable the `allow-import-from-derivation` setting.
- **--allow-new-privileges** Enable the `allow-new-privileges` setting.
- **--allow-symlinked-store** Enable the `allow-symlinked-store` setting.
- **--allow-unsafe-native-code-during-evaluation** Enable the `allow-unsafe-native-code-during-evaluation` setting.
- **--allowed-impure-host-deps** *value* Set the `allowed-impure-host-deps` setting.
- **--allowed-uris** *value* Set the `allowed-uris` setting.
- **--allowed-users** *value* Set the `allowed-users` setting.
- **--auto-allocate-uids** Enable the `auto-allocate-uids` setting.
- **--auto-optimise-store** Enable the `auto-optimise-store` setting.
- **--bash-prompt** *value* Set the `bash-prompt` setting.

- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.
- `--build-hook value` Set the `build-hook` setting.
- `--build-poll-interval value` Set the `build-poll-interval` setting.
- `--build-users-group value` Set the `build-users-group` setting.
- `--builders value` Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.

- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.

- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.

- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.

- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-`

groups setting.

- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.

- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix profile install` - install a package into a profile

# Synopsis

`nix profile install [option...] installables...`

# Examples

- Install a package from Nixpkgs:

```
nix profile install nixpkgs#hello
```

- Install a package from a specific branch of Nixpkgs:

```
nix profile install nixpkgs/release-20.09#hello
```

- Install a package from a specific revision of Nixpkgs:

```
nix profile install
nixpkgs/d73407e8e6002646acfdef0e39ace088bacc83da#hello
```

- Install a specific output of a package:

```
nix profile install nixpkgs#bash^man
```

# Description

This command adds *installables* to a Nix profile.

# Options

- `--priority priority` The priority of the package to install.
- `--profile path` The profile to operate on.
- `--stdin` Read installables from the standard input. No default installable applied.

## Common evaluation options:

- `--arg name expr` Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr name string` Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store store-url` The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include / -I path` Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for nix-channel](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.

- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file` *flake-lock-path* Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input` *input-path flake-url* Override a specific flake input (e.g. `dwarfss/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file` *flake-lock-path* Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input` *input-path* Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.

- **--repair** During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- **--version** Show version information.

## Options that change the interpretation of `installables`:

- **--expr** *expr* Interpret `installables` as attribute paths relative to the Nix expression *expr*.
- **--file** / **-f** *file* Interpret `installables` as attribute paths relative to the Nix expression stored in *file*. If *file* is the character -, then a Nix expression will be read from standard input. Implies **--impure**.

## Options to override configuration settings:

- **--accept-flake-config** Enable the `accept-flake-config` setting.
- **--access-tokens** *value* Set the `access-tokens` setting.
- **--allow-dirty** Enable the `allow-dirty` setting.
- **--allow-import-from-derivation** Enable the `allow-import-from-derivation` setting.
- **--allow-new-privileges** Enable the `allow-new-privileges` setting.
- **--allow-symlinked-store** Enable the `allow-symlinked-store` setting.
- **--allow-unsafe-native-code-during-evaluation** Enable the `allow-unsafe-native-code-during-evaluation` setting.
- **--allowed-impure-host-deps** *value* Set the `allowed-impure-host-deps` setting.
- **--allowed-uris** *value* Set the `allowed-uris` setting.
- **--allowed-users** *value* Set the `allowed-users` setting.
- **--auto-allocate-uids** Enable the `auto-allocate-uids` setting.
- **--auto-optimise-store** Enable the `auto-optimise-store` setting.
- **--bash-prompt** *value* Set the `bash-prompt` setting.
- **--bash-prompt-prefix** *value* Set the `bash-prompt-prefix` setting.
- **--bash-prompt-suffix** *value* Set the `bash-prompt-suffix` setting.

- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.

- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.

- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.

- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.

- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.

- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.

- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix profile list` - list installed packages

# Synopsis

`nix profile list [option...]`

# Examples

- Show what packages are installed in the default profile:

```
nix profile list

Index: 0
Flake attribute: legacyPackages.x86_64-linux.gdb
Original flake URL: flake:nixpkgs
Locked flake URL: github:NixOS/nixpkgs
/7b38b03d76ab71bdc8dc325e3f6338d984cc35ca
Store paths: /nix/store/indzkw5wv1hx6vwk7k4iq29q15chvr3d-gdb-11.1

Index: 1
Flake attribute: packages.x86_64-linux.default
Original flake URL: flake:blender-bin
Locked flake URL: github:edolstra/nix-
warez/91f2ffee657bf834e4475865ae336e2379282d34?dir=blender
Store paths: /nix/store/i798sxl3j40wpdi1rgf391id1b5klw7g-blender-
bin-3.1.2
```

Note that you can unambiguously rebuild a package from a profile through its locked flake URL and flake attribute, e.g.

```
nix build github:edolstra/nix-
warez/91f2ffee657bf834e4475865ae336e2379282d34?dir=blender#packages.x86_64-
linux.default
```

will build the package with index 1 shown above.

## Description

This command shows what packages are currently installed in a profile. For each installed package, it shows the following information:

- **Index** : An integer that can be used to unambiguously identify the package in invocations of `nix profile remove` and `nix profile upgrade`.
- **Flake attribute** : The flake output attribute path that provides the package (e.g. `packages.x86_64-linux.hello`).
- **Original flake URL** : The original ("unlocked") flake reference specified by the user when the package was first installed via `nix profile install`.
- **Locked flake URL** : The locked flake reference to which the original flake reference was resolved.
- **Store paths** : The store path(s) of the package.

## Options

- `--json` Produce output in JSON format, suitable for consumption by another program.
- `--profile path` The profile to operate on.

### Common evaluation options:

- `--arg name expr` Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr name string` Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store store-url` The [URL of the Nix store](#) to use for evaluation, i.e. to store

derivations (`.drv` files) and inputs referenced by them.

- `--impure` Allow access to mutable paths and repositories.
- `--include / -I path` Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.

- **--repair** During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- **--version** Show version information.

## Options to override configuration settings:

- **--accept-flake-config** Enable the `accept-flake-config` setting.
- **--access-tokens** *value* Set the `access-tokens` setting.
- **--allow-dirty** Enable the `allow-dirty` setting.
- **--allow-import-from-derivation** Enable the `allow-import-from-derivation` setting.
- **--allow-new-privileges** Enable the `allow-new-privileges` setting.
- **--allow-symlinked-store** Enable the `allow-symlinked-store` setting.
- **--allow-unsafe-native-code-during-evaluation** Enable the `allow-unsafe-native-code-during-evaluation` setting.
- **--allowed-impure-host-deps** *value* Set the `allowed-impure-host-deps` setting.
- **--allowed-uris** *value* Set the `allowed-uris` setting.
- **--allowed-users** *value* Set the `allowed-users` setting.
- **--auto-allocate-uids** Enable the `auto-allocate-uids` setting.
- **--auto-optimise-store** Enable the `auto-optimise-store` setting.
- **--bash-prompt** *value* Set the `bash-prompt` setting.
- **--bash-prompt-prefix** *value* Set the `bash-prompt-prefix` setting.
- **--bash-prompt-suffix** *value* Set the `bash-prompt-suffix` setting.
- **--build-hook** *value* Set the `build-hook` setting.
- **--build-poll-interval** *value* Set the `build-poll-interval` setting.
- **--build-users-group** *value* Set the `build-users-group` setting.
- **--builders** *value* Set the `builders` setting.

- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors value` Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls value` Append to the `ignored-acls` setting.
- `--extra-nix-path value` Append to the `nix-path` setting.
- `--extra-platforms value` Set the `extra-platforms` setting.
- `--extra-plugin-files value` Append to the `plugin-files` setting.
- `--extra-sandbox-paths value` Append to the `sandbox-paths` setting.

- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.

- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.

- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.

- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.

- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.

- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix profile remove` - remove packages from a profile

# Synopsis

`nix profile remove [option...] elements...`

# Examples

- Remove a package by position:

```
nix profile remove 3
```

- Remove a package by attribute path:

```
nix profile remove packages.x86_64-linux.hello
```

- Remove all packages:

```
nix profile remove '.*'
```

- Remove a package by store path:

```
nix profile remove /nix/store/rr3y0c6zyk7kjjl8y19s4lsrhn4aiq1z-hello-2.10
```

# Description

This command removes a package from a profile.

# Options

- `--profile path` The profile to operate on.

## Common evaluation options:

- `--arg name expr` Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr name string` Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store store-url` The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include / -I path` Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.

- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps value` Set the `allowed-impure-host-deps` setting.
- `--allowed-uris value` Set the `allowed-uris` setting.
- `--allowed-users value` Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.

- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt value` Set the `bash-prompt` setting.
- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.
- `--build-hook value` Set the `build-hook` setting.
- `--build-poll-interval value` Set the `build-poll-interval` setting.
- `--build-users-group value` Set the `build-users-group` setting.
- `--builders value` Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.

- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.

- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls value` Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.

- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.

- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.

- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.

- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

**Warning**

This program is **experimental** and its interface is subject to change.

---

# Name

`nix profile rollback` - roll back to the previous version or a specified version of a profile

# Synopsis

`nix profile rollback [option...]`

# Examples

- Roll back your default profile to the previous version:

```
nix profile rollback
switching profile from version 519 to 518
```

- Switch your default profile to version 510:

```
nix profile rollback --to 510
switching profile from version 518 to 510
```

# Description

This command switches a profile to the most recent version older than the currently active version, or if `--to N` is given, to version `N` of the profile. To see the available versions of a profile, use `nix profile history`.

# Options

- `--dry-run` Show what this command would do without doing it.
- `--profile path` The profile to operate on.
- `--to version` The profile version to roll back to.

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.

- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps value` Set the `allowed-impure-host-deps` setting.
- `--allowed-uris value` Set the `allowed-uris` setting.
- `--allowed-users value` Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt value` Set the `bash-prompt` setting.
- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.
- `--build-hook value` Set the `build-hook` setting.
- `--build-poll-interval value` Set the `build-poll-interval` setting.
- `--build-users-group value` Set the `build-users-group` setting.
- `--builders value` Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.

- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.

- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space value` Set the `gc-reserved-space` setting.
- `--hashed-mirrors value` Set the `hashed-mirrors` setting.
- `--http-connections value` Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count value` Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls value` Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.

- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.

- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.

- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.
- `--start-id value` Set the `start-id` setting.
- `--store value` Set the `store` setting.

- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix profile upgrade` - upgrade packages using their most recent flake

# Synopsis

`nix profile upgrade [option...] elements...`

# Examples

- Upgrade all packages that were installed using an unlocked flake reference:

```
nix profile upgrade '.★'
```

- Upgrade a specific package:

```
nix profile upgrade packages.x86_64-linux.hello
```

- Upgrade a specific profile element by number:

```
nix profile list
0 flake:nixpkgs#legacyPackages.x86_64-linux.spotify ...
nix profile upgrade 0
```

# Description

This command upgrades a previously installed package in a Nix profile, by fetching and evaluating the latest version of the flake from which the package was installed.

---

## Warning

This only works if you used an *unlocked* flake reference at installation time, e.g. `nixpkgs#hello`. It does not work if you used a *locked* flake reference (e.g. `github:NixOS/nixpkgs/13d0c311e3ae923a00f734b43fd1d35b47d8943a#hello`), since in that case the "latest version" is always the same.

---

# Options

- `--profile path` The profile to operate on.

## Common evaluation options:

- `--arg name expr` Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr name string` Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store store-url` The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include / -I path` Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for nix-channel](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding

`nix.conf`).

- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options that change the interpretation of `installables`:

- `--expr expr` Interpret `installables` as attribute paths relative to the Nix expression `expr`.
- `--file / -f file` Interpret `installables` as attribute paths relative to the Nix expression stored in `file`. If `file` is the character `-`, then a Nix expression will be read from standard input. Implies `--impure`.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps value` Set the `allowed-impure-host-deps` setting.
- `--allowed-uris value` Set the `allowed-uris` setting.
- `--allowed-users value` Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt value` Set the `bash-prompt` setting.

- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.
- `--build-hook value` Set the `build-hook` setting.
- `--build-poll-interval value` Set the `build-poll-interval` setting.
- `--build-users-group value` Set the `build-users-group` setting.
- `--builders value` Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.

- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.

- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.

- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.

- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-`

groups setting.

- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.

- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix profile wipe-history` - delete non-current versions of a profile

# Synopsis

`nix profile wipe-history [option...]`

# Examples

- Delete all versions of the default profile older than 100 days:

```
nix profile wipe-history --profile /tmp/profile --older-than 100d
removing profile version 515
removing profile version 514
```

# Description

This command deletes non-current versions of a profile, making it impossible to roll back to these versions. By default, all non-current versions are deleted. With `--older-than N d`, all non-current versions older than  $N$  days are deleted.

# Options

- `--dry-run` Show what this command would do without doing it.
- `--older-than age` Delete versions older than the specified age. *age* must be in the format  $N d$ , where  $N$  denotes a number of days.

- `--profile path` The profile to operate on.

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.

- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps`

setting.

- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.

- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.
- `--max-silent-time` *value* Set the `max-silent-time` setting.
- `--max-substitution-jobs` *value* Set the `max-substitution-jobs` setting.
- `--min-free` *value* Set the `min-free` setting.
- `--min-free-check-interval` *value* Set the `min-free-check-interval` setting.
- `--nar-buffer-size` *value* Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl` *value* Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl` *value* Set the `narinfo-cache-positive-ttl` setting.

- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.

- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.

- `--post-build-hook` *value* Set the `post-build-hook` setting.
- `--pre-build-hook` *value* Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.

- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

**Warning**

This program is **experimental** and its interface is subject to change.

---

# Name

`nix realisation` - manipulate a Nix realisation

# Synopsis

`nix realisation [option...] subcommand`

where *subcommand* is one of the following:

- `nix realisation info` - query information about one or several realisations

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix realisation info` - query information about one or several realisations

## Synopsis

`nix realisation info [option...] installables...`

## Description

Display some information about the given realisation

## Examples

Show some information about the realisation of the `hello` package:

```
$ nix realisation info nixpkgs#hello --json
[{"id":"sha256:3d382378a00588e064ee30be96dd0fa7e7df7cf3fbcae85a0e7b7dada1eef25
!out","outPath":"fd3m7xawvrqcg98kgz5hc2vk3x9q0lh7-hello"}]
```

## Options

- `--json` Produce output in JSON format, suitable for consumption by another program.
- `--stdin` Read installables from the standard input. No default installable applied.

### Common evaluation options:

- `--arg name expr` Pass the value `expr` as the argument `name` to Nix functions.

- **--argstr** *name string* Pass the string *string* as the argument *name* to Nix functions.
- **--debugger** Start an interactive environment if evaluation fails.
- **--eval-store** *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- **--impure** Allow access to mutable paths and repositories.
- **--include** / **-I** *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for](#)

`nix-channel`) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.

- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

### Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

### Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

### Options that change the interpretation of `installables`:

- `--all` Apply the operation to every store path.
- `--derivation` Operate on the `store derivation` rather than its outputs.
- `--expr expr` Interpret `installables` as attribute paths relative to the Nix expression `expr`.
- `--file / -f file` Interpret `installables` as attribute paths relative to the Nix expression stored in `file`. If `file` is the character `-`, then a Nix expression will be read from standard

input. Implies `--impure`.

- `--recursive` / `-r` Apply operation to closure of the specified paths.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.

- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors value` Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls value` Append to the `ignored-acls` setting.
- `--extra-nix-path value` Append to the `nix-path` setting.
- `--extra-platforms value` Set the `extra-platforms` setting.
- `--extra-plugin-files value` Append to the `plugin-files` setting.
- `--extra-sandbox-paths value` Append to the `sandbox-paths` setting.
- `--extra-secret-key-files value` Append to the `secret-key-files` setting.

- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.

- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.

- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.

- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.

- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.

- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix registry` - manage the flake registry

## Synopsis

`nix registry [option...] subcommand`

where *subcommand* is one of the following:

- `nix registry add` - add/replace flake in user flake registry
- `nix registry list` - list available Nix flakes
- `nix registry pin` - pin a flake to its current version or to the current version of a flake URL
- `nix registry remove` - remove flake from user flake registry

## Description

`nix registry` provides subcommands for managing *flake registries*. Flake registries are a convenience feature that allows you to refer to flakes using symbolic identifiers such as `nixpkgs`, rather than full URLs such as `git://github.com/NixOS/nixpkgs`. You can use these identifiers on the command line (e.g. when you do `nix run nixpkgs#hello`) or in flake input specifications in `flake.nix` files. The latter are automatically resolved to full URLs and recorded in the flake's `flake.lock` file.

In addition, the flake registry allows you to redirect arbitrary flake references (e.g. `github:NixOS/patchelf`) to another location, such as a local fork.

There are multiple registries. These are, in order from lowest to highest precedence:

- The global registry, which is a file downloaded from the URL specified by the setting `flake-registry`. It is cached locally and updated automatically when it's older than

`tarball-ttl` seconds. The default global registry is kept in a [GitHub repository](#).

- The system registry, which is shared by all users. The default location is `/etc/nix/registry.json`. On NixOS, the system registry can be specified using the NixOS option `nix.registry`.
- The user registry `~/.config/nix/registry.json`. This registry can be modified by commands such as `nix registry pin`.
- Overrides specified on the command line using the option `--override-flake`.

## Registry format

A registry is a JSON file with the following format:

```
{
 "version": 2,
 "flakes": [
 {
 "from": {
 "type": "indirect",
 "id": "nixpkgs"
 },
 "to": {
 "type": "github",
 "owner": "NixOS",
 "repo": "nixpkgs"
 }
 },
 ...
]
}
```

That is, it contains a list of objects with attributes `from` and `to`, both of which contain a flake reference in attribute representation. (For example, `{"type": "indirect", "id": "nixpkgs"}` is the attribute representation of `nixpkgs`, while `{"type": "github", "owner": "NixOS", "repo": "nixpkgs"}` is the attribute representation of `github:NixOS/nixpkgs`.)

Given some flake reference  $R$ , a registry entry is used if its `from` flake reference *matches*  $R$ .  $R$  is then replaced by the *unification* of the `to` flake reference with  $R$ .

# Matching

The `from` flake reference in a registry entry *matches* some flake reference  $R$  if the attributes in `from` are the same as the attributes in  $R$ . For example:

- `nixpkgs` matches with `nixpkgs`.
- `nixpkgs` matches with `nixpkgs/nixos-20.09`.
- `nixpkgs/nixos-20.09` does not match with `nixpkgs`.
- `nixpkgs` does not match with `git://github.com/NixOS/patchelf`.

# Unification

The `to` flake reference in a registry entry is *unified* with some flake reference  $R$  by taking `to` and applying the `rev` and `ref` attributes from  $R$ , if specified. For example:

- `github:NixOS/nixpkgs` unified with `nixpkgs` produces `github:NixOS/nixpkgs`.
- `github:NixOS/nixpkgs` unified with `nixpkgs/nixos-20.09` produces `github:NixOS/nixpkgs/nixos-20.09`.
- `github:NixOS/nixpkgs/master` unified with `nixpkgs/nixos-20.09` produces `github:NixOS/nixpkgs/nixos-20.09`.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix registry add` - add/replace flake in user flake registry

## Synopsis

`nix registry add [option...] from-url to-url`

# Examples

- Set the `nixpkgs` flake identifier to a specific branch of Nixpkgs:

```
nix registry add nixpkgs github:NixOS/nixpkgs/nixos-20.03
```

- Pin `nixpkgs` to a specific revision:

```
nix registry add nixpkgs github:NixOS/nixpkgs
/925b70cd964ceaedee26fde9b19cc4c4f081196a
```

- Add an entry that redirects a specific branch of `nixpkgs` to another fork:

```
nix registry add nixpkgs/nixos-20.03 ~/Dev/nixpkgs
```

- Add `nixpkgs` pointing to `github:nixos/nixpkgs` to your custom flake registry:

```
nix registry add --registry ./custom-flake-registry.json nixpkgs
github:nixos/nixpkgs
```

# Description

This command adds an entry to the user registry that maps flake reference *from-url* to flake reference *to-url*. If an entry for *from-url* already exists, it is overwritten.

Entries can be removed using `nix registry remove`.

# Options

- `--registry` *registry* The registry to operate on.

## Common evaluation options:

- `--arg` *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr` *name string* Pass the string *string* as the argument *name* to Nix functions.
- `--eval-store` *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include` / `-I` *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for nix-channel](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.

- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps`

setting.

- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.

- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.
- `--max-silent-time` *value* Set the `max-silent-time` setting.
- `--max-substitution-jobs` *value* Set the `max-substitution-jobs` setting.
- `--min-free` *value* Set the `min-free` setting.
- `--min-free-check-interval` *value* Set the `min-free-check-interval` setting.
- `--nar-buffer-size` *value* Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl` *value* Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl` *value* Set the `narinfo-cache-positive-ttl` setting.

- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.

- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.

- `--post-build-hook` *value* Set the `post-build-hook` setting.
- `--pre-build-hook` *value* Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.

- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix registry list` - list available Nix flakes

# Synopsis

`nix registry list [option...]`

# Examples

- Show the contents of all registries:

```
nix registry list
user flake:dwarfss github:edolstra/dwarfss
/d181d714fd36eb06f4992a1997cd5601e26db8f5
system flake:nixpkgs path:/nix/store/fxl9mrm5xvzam0lx19ygdmksskx4qq8s-
source?lastModified=1605220118&narHash=sha256-
Und10ixH1WuW0XHYMxxuHRohKYb45R%2fT8CwZuLd2D2Q=&
rev=3090c65041104931adda7625d37fa874b2b5c124
global flake:blender-bin github:edolstra/nix-warez?dir=blender
global flake:dwarfss github:edolstra/dwarfss
...
...
```

# Description

This command displays the contents of all registries on standard output. Each line represents one registry entry in the format *type from to*, where *type* denotes the registry containing the entry:

- `flags` : entries specified on the command line using `--override-flake` .
- `user` : the user registry.
- `system` : the system registry.
- `global` : the global registry.

See the [nix registry manual page](#) for more details.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix registry pin` - pin a flake to its current version or to the current version of a flake URL

## Synopsis

`nix registry pin [option...] url locked`

# Examples

- Pin `nixpkgs` to its most recent Git revision:

```
nix registry pin nixpkgs
```

Afterwards the user registry will have an entry like this:

```
nix registry list | grep '^user '
user flake:nixpkgs github:NixOS/nixpkgs
/925b70cd964ceaedee26fde9b19cc4c4f081196a
```

and `nix flake info` will say:

```
nix flake info nixpkgs
Resolved URL: github:NixOS/nixpkgs
/925b70cd964ceaedee26fde9b19cc4c4f081196a
Locked URL: github:NixOS/nixpkgs
/925b70cd964ceaedee26fde9b19cc4c4f081196a
...
...
```

- Pin `nixpkgs` in a custom registry to its most recent Git revision:

```
nix registry pin --registry ./custom-flake-registry.json nixpkgs
```

# Description

This command adds an entry to the user registry that maps flake reference *url* to the corresponding *locked* flake reference, that is, a flake reference that specifies an exact revision or content hash. This ensures that until this registry entry is removed, all uses of *url* will resolve to exactly the same flake.

Entries can be removed using `nix registry remove`.

# Options

- `--registry` *registry* The registry to operate on.

## Common evaluation options:

- `--arg` *name* *expr* Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr` *name* *string* Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store` *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include` / `-I` *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for nix-channel](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting

*original-ref* to *resolved-ref*.

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.

- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps value` Set the `allowed-impure-host-deps` setting.
- `--allowed-uris value` Set the `allowed-uris` setting.
- `--allowed-users value` Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt value` Set the `bash-prompt` setting.
- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.
- `--build-hook value` Set the `build-hook` setting.
- `--build-poll-interval value` Set the `build-poll-interval` setting.
- `--build-users-group value` Set the `build-users-group` setting.
- `--builders value` Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.

- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.

- `--gc-reserved-space value` Set the `gc-reserved-space` setting.
- `--hashed-mirrors value` Set the `hashed-mirrors` setting.
- `--http-connections value` Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count value` Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls value` Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.

- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.

- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.

- `--plugin-files` *value* Set the `plugin-files` setting.
- `--post-build-hook` *value* Set the `post-build-hook` setting.
- `--pre-build-hook` *value* Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.

- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix registry remove` - remove flake from user flake registry

# Synopsis

`nix registry remove [option...] url`

# Examples

- Remove the entry `nixpkgs` from the user registry:

```
nix registry remove nixpkgs
```

- Remove the entry `nixpkgs` from a custom registry:

```
nix registry remove --registry ./custom-flake-registry.json nixpkgs
```

# Description

This command removes from the user registry any entry for flake reference *url*.

# Options

- `--registry` *registry* The registry to operate on.

**Logging-related options:**

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.

- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.

- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.

- `--http2` Enable the `http2` setting.
- `--id-count value` Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls value` Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.

- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.

- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.

- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.

- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

**Warning**

This program is **experimental** and its interface is subject to change.

---

# Name

`nix repl` - start an interactive environment for evaluating Nix expressions

## Synopsis

`nix repl [option...] installables...`

# Examples

- Display all special commands within the REPL:

```
nix repl
nix-repl> ?:
```

- Evaluate some simple Nix expressions:

```
nix repl

nix-repl> 1 + 2
3

nix-repl> map (x: x * 2) [1 2 3]
[2 4 6]
```

- Interact with Nixpkgs in the REPL:

```
nix repl --file example.nix
Loading Installable ''...
Added 3 variables.

nix repl --expr '{a={b=3;c=4;};}'
Loading Installable ''...
Added 1 variables.

nix repl --expr '{a={b=3;c=4;};}' a
Loading Installable ''...
Added 1 variables.

nix repl --extra-experimental-features 'flakes repl-flake' nixpkgs
Loading Installable 'flake:nixpkgs#...'
Added 5 variables.

nix-repl> legacyPackages.x86_64-linux.emacs.name
"emacs-27.1"

nix-repl> legacyPackages.x86_64-linux.emacs.name
"emacs-27.1"

nix-repl> :q

nix repl --expr 'import <nixpkgs>{}'
Loading Installable ''...
Added 12439 variables.

nix-repl> emacs.name
"emacs-27.1"

nix-repl> emacsdrvPath
"/nix/store/lp0sjrhgg03y2n0l10n70rg0k7hhyz0l-emacs-27.1.drv"

nix-repl> drv = runCommand "hello" { buildInputs = [hello]; } "hello;
hello > $out"

nix-repl> :b drv
this derivation produced the following outputs:
```

```
out -> /nix/store/0njwbgwmkwls0w5dv9mpc1pq5fj39q0l-hello

nix-repl> builtins.readFile drv
"Hello, world!\n"

nix-repl> :log drv
Hello, world!
```

# Description

This command provides an interactive environment for evaluating Nix expressions. (REPL stands for 'read–eval–print loop').

On startup, it loads the Nix expressions named *files* and adds them into the lexical scope. You can load addition files using the `:l <filename>` command, or reload all files using `:r .`

# Options

- `--stdin` Read installables from the standard input. No default installable applied.

## Common evaluation options:

- `--arg name expr` Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr name string` Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store store-url` The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include / -I path` Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>` ).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.

- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options that change the interpretation of `installables`:

- `--expr` *expr* Interpret `installables` as attribute paths relative to the Nix expression *expr*.
- `--file` / `-f` *file* Interpret `installables` as attribute paths relative to the Nix expression stored in *file*. If *file* is the character -, then a Nix expression will be read from standard input. Implies `--impure`.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.

- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.

- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.

- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.
- `--max-silent-time` *value* Set the `max-silent-time` setting.
- `--max-substitution-jobs` *value* Set the `max-substitution-jobs` setting.
- `--min-free` *value* Set the `min-free` setting.
- `--min-free-check-interval` *value* Set the `min-free-check-interval` setting.
- `--nar-buffer-size` *value* Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl` *value* Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl` *value* Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file` *value* Set the `netrc-file` setting.

- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.

- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.

- `--pre-build-hook` *value* Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.

- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix run` - run a Nix application

## Synopsis

`nix run [option...] installable args...`

# Examples

- Run the default app from the `blender-bin` flake:

```
nix run blender-bin
```

- Run a non-default app from the `blender-bin` flake:

```
nix run blender-bin#blender_2_83
```

Tip: you can find apps provided by this flake by running `nix flake show blender-bin`.

- Run `vim` from the `nixpkgs` flake:

```
nix run nixpkgs#vim
```

Note that `vim` (as of the time of writing of this page) is not an app but a package. Thus, Nix runs the eponymous file from the `vim` package.

- Run `vim` with arguments:

```
nix run nixpkgs#vim -- --help
```

# Description

`nix run` builds and runs *installable*, which must evaluate to an *app* or a regular Nix derivation.

If *installable* evaluates to an *app* (see below), it executes the program specified by the app definition.

If *installable* evaluates to a derivation, it will try to execute the program `<out>/bin/<name>`, where *out* is the primary output store path of the derivation, and *name* is the first of the following that exists:

- The `meta.mainProgram` attribute of the derivation.
- The `pname` attribute of the derivation.
- The name part of the value of the `name` attribute of the derivation.

For instance, if `name` is set to `hello-1.10`, `nix run` will run `$out/bin/hello`.

# Flake output attributes

If no flake output attribute is given, `nix run` tries the following flake output attributes:

- `apps.<system>.default`
- `packages.<system>.default`

If an attribute *name* is given, `nix run` tries the following flake output attributes:

- `apps.<system>.<name>`
- `packages.<system>.<name>`
- `legacyPackages.<system>.<name>`

# Apps

An app is specified by a flake output attribute named `apps.<system>.<name>`. It looks like this:

```
apps.x86_64-linux.blender_2_79 = {
 type = "app";
 program = "${self.packages.x86_64-linux.blender_2_79}/bin/blender";
};
```

The only supported attributes are:

- `type` (required): Must be set to `app`.
- `program` (required): The full path of the executable to run. It must reside in the Nix store.

# Options

## Common evaluation options:

- `--arg` *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr` *name string* Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store` *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include` / `-I` *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarfss/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.

- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options that change the interpretation of `installables`:

- `--expr expr` Interpret `installables` as attribute paths relative to the Nix expression `expr`.
- `--file / -f file` Interpret `installables` as attribute paths relative to the Nix expression stored in `file`. If `file` is the character `-`, then a Nix expression will be read from standard input. Implies `--impure`.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps value` Set the `allowed-impure-host-deps` setting.
- `--allowed-uris value` Set the `allowed-uris` setting.
- `--allowed-users value` Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.

- `--bash-prompt value` Set the `bash-prompt` setting.
- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.
- `--build-hook value` Set the `build-hook` setting.
- `--build-poll-interval value` Set the `build-poll-interval` setting.
- `--build-users-group value` Set the `build-users-group` setting.
- `--builders value` Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.

- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.

- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.

- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.

- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.

- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.

- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix search` - search for packages

## Synopsis

`nix search [option...] installable regex...`

## Examples

- Show all packages in the `nixpkgs` flake:

```
nix search nixpkgs
* legacyPackages.x86_64-linux.AMB-plugins (0.8.1)
 A set of ambisonics ladspa plugins

* legacyPackages.x86_64-linux.ArchiSteamFarm (4.3.1.0)
 Application with primary purpose of idling Steam cards from multiple
 accounts simultaneously
...
```

- Show packages in the `nixpkgs` flake containing `blender` in its name or description:

```
nix search nixpkgs blender
* legacyPackages.x86_64-linux.blender (2.91.0)
 3D Creation/Animation/Publishing System
```

- Search for packages underneath the attribute `gnome3` in Nixpkgs:

```
nix search nixpkgs#gnome3 vala
* legacyPackages.x86_64-linux.gnome3.vala (0.48.9)
 Compiler for GObject type system
```

- Show all packages in the flake in the current directory:

```
nix search
```

- Search for Firefox or Chromium:

```
nix search nixpkgs 'firefox|chromium'
```

- Search for packages containing `git` and either `frontend` or `gui`:

```
nix search nixpkgs git 'frontend|gui'
```

- Search for packages containing `neovim` but hide ones containing either `gui` or `python`:

```
nix search nixpkgs neovim --exclude 'python|gui'
```

or

```
nix search nixpkgs neovim --exclude 'python' --exclude 'gui'
```

## Description

`nix search` searches `installable` (which can be evaluated, that is, a flake or Nix expression, but not a store path or store derivation path) for packages whose name or description matches all of the regular expressions `regex`. For each matching package, it prints the full attribute name (from the root of the `installable`), the version and the `meta.description` field, highlighting the substrings that were matched by the regular expressions. If no regular expressions are specified, all packages are shown.

## Flake output attributes

If no flake output attribute is given, `nix search` searches for packages:

- Directly underneath `packages.<system>`.
- Underneath `legacyPackages.<system>`, recursing into attribute sets that contain an attribute `reurseForDerivations = true`.

# Options

- `--exclude / -e regex` Hide packages whose attribute path, name or description contain `regex`.
- `--json` Produce output in JSON format, suitable for consumption by another program.

## Common evaluation options:

- `--arg name expr` Pass the value `expr` as the argument `name` to Nix functions.
- `--argstr name string` Pass the string `string` as the argument `name` to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store store-url` The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include / -I path` Add `path` to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for nix-channel](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarfss/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.

- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options that change the interpretation of `installables`:

- `--expr expr` Interpret `installables` as attribute paths relative to the Nix expression `expr`.
- `--file / -f file` Interpret `installables` as attribute paths relative to the Nix expression stored in `file`. If `file` is the character `-`, then a Nix expression will be read from standard input. Implies `--impure`.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps value` Set the `allowed-impure-host-deps` setting.
- `--allowed-uris value` Set the `allowed-uris` setting.
- `--allowed-users value` Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.

- `--bash-prompt value` Set the `bash-prompt` setting.
- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.
- `--build-hook value` Set the `build-hook` setting.
- `--build-poll-interval value` Set the `build-poll-interval` setting.
- `--build-users-group value` Set the `build-users-group` setting.
- `--builders value` Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.

- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.

- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.

- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.

- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.

- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.

- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix shell` - run a shell in which the specified packages are available

## Synopsis

`nix shell [option...] installables...`

# Examples

- Start a shell providing `youtube-dl` from the `nixpkgs` flake:

```
nix shell nixpkgs#youtube-dl
youtube-dl --version
2020.11.01.1
```

- Start a shell providing GNU Hello from NixOS 20.03:

```
nix shell nixpkgs/nixos-20.03#hello
```

- Run GNU Hello:

```
nix shell nixpkgs#hello --command hello --greeting 'Hi everybody!'
Hi everybody!
```

- Run multiple commands in a shell environment:

```
nix shell nixpkgs#gnumake --command sh -c "cd src && make"
```

- Run GNU Hello in a chroot store:

```
nix shell --store ~/my-nix nixpkgs#hello --command hello
```

- Start a shell providing GNU Hello in a chroot store:

```
nix shell --store ~/my-nix nixpkgs#hello nixpkgs#bashInteractive
--command bash
```

Note that it's necessary to specify `bash` explicitly because your default shell (e.g. `/bin/bash`) generally will not exist in the chroot.

## Description

`nix shell` runs a command in an environment in which the `$PATH` variable provides the specified *installables*. If no command is specified, it starts the default shell of your user account specified by `$SHELL`.

## Options

- `--command` / `-c` *command args* Command and arguments to be executed, defaulting to `$SHELL`
- `--ignore-environment` / `-i` Clear the entire environment (except those specified with `--keep`).
- `--keep` / `-k` *name* Keep the environment variable *name*.
- `--stdin` Read installables from the standard input. No default installable applied.
- `--unset` / `-u` *name* Unset the environment variable *name*.

### Common evaluation options:

- `--arg` *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr` *name string* Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store` *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store

derivations (`.drv` files) and inputs referenced by them.

- `--impure` Allow access to mutable paths and repositories.
- `--include / -I path` Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.

- `--update-input` *input-path* Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options that change the interpretation of `installables`:

- `--expr` *expr* Interpret `installables` as attribute paths relative to the Nix expression *expr*.
- `--file` / `-f` *file* Interpret `installables` as attribute paths relative to the Nix expression stored in *file*. If *file* is the character `-`, then a Nix expression will be read from standard input. Implies `--impure`.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.

- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.

- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.

- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.
- `--max-silent-time` *value* Set the `max-silent-time` setting.
- `--max-substitution-jobs` *value* Set the `max-substitution-jobs` setting.

- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.

- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.

- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.

- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix show-config` - show the Nix configuration or the value of a specific setting

## Synopsis

`nix show-config [option...] name`

# Options

- `--json` Produce output in JSON format, suitable for consumption by another program.

### Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

### Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).

- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.

- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors value` Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls value` Append to the `ignored-acls` setting.
- `--extra-nix-path value` Append to the `nix-path` setting.
- `--extra-platforms value` Set the `extra-platforms` setting.
- `--extra-plugin-files value` Append to the `plugin-files` setting.
- `--extra-sandbox-paths value` Append to the `sandbox-paths` setting.
- `--extra-secret-key-files value` Append to the `secret-key-files` setting.

- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.

- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.

- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.

- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.

- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.

- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

**Warning**

This program is **experimental** and its interface is subject to change.

---

# Name

`nix store` - manipulate a Nix store

# Synopsis

`nix store [option...] subcommand`

where *subcommand* is one of the following:

- `nix store add-file` - add a regular file to the Nix store
- `nix store add-path` - add a path to the Nix store
- `nix store cat` - print the contents of a file in the Nix store on stdout
- `nix store copy-log` - copy build logs between Nix stores
- `nix store copy-sigs` - copy store path signatures from substituters
- `nix store delete` - delete paths from the Nix store
- `nix store diff-closures` - show what packages and versions were added and removed between two closures
- `nix store dump-path` - serialise a store path to stdout in NAR format
- `nix store gc` - perform garbage collection on a Nix store
- `nix store ls` - show information about a path in the Nix store
- `nix store make-content-addressed` - rewrite a path or closure to content-addressed form
- `nix store optimise` - replace identical files in the store by hard links
- `nix store path-from-hash-part` - get a store path from its hash part
- `nix store ping` - test whether a store can be accessed
- `nix store prefetch-file` - download a file into the Nix store
- `nix store repair` - repair store paths
- `nix store sign` - sign store paths
- `nix store verify` - verify the integrity of store paths

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix store add-file` - add a regular file to the Nix store

## Synopsis

`nix store add-file [option...] path`

## Description

Copy the regular file *path* to the Nix store, and print the resulting store path on standard output.

---

## Warning

The resulting store path is not registered as a garbage collector root, so it could be deleted before you have a chance to register it.

---

## Examples

Add a regular file to the store:

```
echo foo > bar

nix store add-file ./bar
/nix/store/cbv2s4bsvzjri77s2gb8g8bpcb6dpa8w-bar

cat /nix/store/cbv2s4bsvzjri77s2gb8g8bpcb6dpa8w-bar
foo
```

# Options

- `--dry-run` Show what this command would do without doing it.
- `--name / -n name` Override the name component of the store path. It defaults to the base name of *path*.

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.

- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.

- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.

- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.
- `--max-silent-time` *value* Set the `max-silent-time` setting.
- `--max-substitution-jobs` *value* Set the `max-substitution-jobs` setting.
- `--min-free` *value* Set the `min-free` setting.
- `--min-free-check-interval` *value* Set the `min-free-check-interval` setting.

- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.

- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.

- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.
- `--start-id value` Set the `start-id` setting.

- `--store value` Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix store add-path` - add a path to the Nix store

# Synopsis

`nix store add-path [option...] path`

# Description

Copy *path* to the Nix store, and print the resulting store path on standard output.

---

## Warning

The resulting store path is not registered as a garbage collector root, so it could be deleted before you have a chance to register it.

---

# Examples

Add a directory to the store:

```
mkdir dir
echo foo > dir/bar

nix store add-path ./dir
/nix/store/6pmjx56pm94n66n4qw1nff0y1crm8nqg-dir

cat /nix/store/6pmjx56pm94n66n4qw1nff0y1crm8nqg-dir/bar
foo
```

# Options

- `--dry-run` Show what this command would do without doing it.
- `--name / -n name` Override the name component of the store path. It defaults to the base name of *path*.

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.

- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.

- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.

- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.
- `--max-silent-time` *value* Set the `max-silent-time` setting.
- `--max-substitution-jobs` *value* Set the `max-substitution-jobs` setting.
- `--min-free` *value* Set the `min-free` setting.
- `--min-free-check-interval` *value* Set the `min-free-check-interval` setting.

- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.

- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.

- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.
- `--start-id value` Set the `start-id` setting.

- `--store value` Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix store cat` - print the contents of a file in the Nix store on stdout

# Synopsis

`nix store cat [option...] path`

# Examples

- Show the contents of a file in a binary cache:

```
nix store cat --store https://cache.nixos.org/ \
/nix/store/0i2jd68mp5g6h2sa5k9c85rb80sn8hi9-hello-2.10/bin/hello |
hexdump -C | head -n1
00000000 7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
|.ELF.....|
```

# Description

This command prints on standard output the contents of the regular file *path* in a Nix store. *path* can be a top-level store path or any file inside a store path.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix store copy-log` - copy build logs between Nix stores

# Synopsis

`nix store copy-log [option...] installables...`

# Examples

- To copy the build log of the `hello` package from <https://cache.nixos.org> to the local store:

```
nix store copy-log --from https://cache.nixos.org --eval-store auto
nixpkgs#hello
```

You can verify that the log is available locally:

```
nix log --substituters '' nixpkgs#hello
```

(The flag `--substituters ''` avoids querying <https://cache.nixos.org> for the log.)

- To copy the log for a specific **store derivation** via SSH:

```
nix store copy-log --to ssh-ng://machine /nix/store
/ilgm50plpmcgjhcp33z6n4qbnpqfhxym-glibc-2.33-59.drv
```

# Description

`nix store copy-log` copies build logs between two Nix stores. The source store is specified using `--from` and the destination using `--to`. If one of these is omitted, it defaults to the local store.

# Options

- `--from store-uri` URL of the source Nix store.
- `--stdin` Read installables from the standard input. No default installable applied.
- `--to store-uri` URL of the destination Nix store.

## Common evaluation options:

- `--arg name expr` Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr name string` Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store store-url` The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include / -I path` Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for nix-channel](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding

`nix.conf`).

- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options that change the interpretation of `installables`:

- `--expr expr` Interpret `installables` as attribute paths relative to the Nix expression `expr`.
- `--file / -f file` Interpret `installables` as attribute paths relative to the Nix expression stored in `file`. If `file` is the character `-`, then a Nix expression will be read from standard input. Implies `--impure`.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps value` Set the `allowed-impure-host-deps` setting.
- `--allowed-uris value` Set the `allowed-uris` setting.
- `--allowed-users value` Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt value` Set the `bash-prompt` setting.

- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.
- `--build-hook value` Set the `build-hook` setting.
- `--build-poll-interval value` Set the `build-poll-interval` setting.
- `--build-users-group value` Set the `build-users-group` setting.
- `--builders value` Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.

- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.

- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.

- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.

- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-`

groups setting.

- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.

- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

**Warning**

This program is **experimental** and its interface is subject to change.

---

# Name

`nix store copy-sigs` - copy store path signatures from substituters

## Synopsis

`nix store copy-sigs [option...] installables...`

# Options

- `--stdin` Read installables from the standard input. No default installable applied.
- `--substituter / -s store-uri` Copy signatures from the specified store.

### Common evaluation options:

- `--arg name expr` Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr name string` Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store store-url` The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include / -I path` Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.

- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options that change the interpretation of `installables`:

- `--all` Apply the operation to every store path.
- `--derivation` Operate on the `store derivation` rather than its outputs.
- `--expr` *expr* Interpret `installables` as attribute paths relative to the Nix expression *expr*.
- `--file` / `-f` *file* Interpret `installables` as attribute paths relative to the Nix expression stored in *file*. If *file* is the character `-`, then a Nix expression will be read from standard input. Implies `--impure`.
- `--recursive` / `-r` Apply operation to closure of the specified paths.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.

- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps value` Set the `allowed-impure-host-deps` setting.
- `--allowed-uris value` Set the `allowed-uris` setting.
- `--allowed-users value` Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt value` Set the `bash-prompt` setting.
- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.
- `--build-hook value` Set the `build-hook` setting.
- `--build-poll-interval value` Set the `build-poll-interval` setting.
- `--build-users-group value` Set the `build-users-group` setting.
- `--builders value` Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.

- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.

- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space value` Set the `gc-reserved-space` setting.
- `--hashed-mirrors value` Set the `hashed-mirrors` setting.
- `--http-connections value` Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count value` Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls value` Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.

- `--narinfo-cache-negative-ttl` *value* Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl` *value* Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file` *value* Set the `netrc-file` setting.
- `--nix-path` *value* Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.

- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.

- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.
- `--start-id value` Set the `start-id` setting.
- `--store value` Set the `store` setting.

- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix store delete` - delete paths from the Nix store

# Synopsis

`nix store delete [option...] installables...`

# Examples

- Delete a specific store path:

```
nix store delete /nix/store/yb5q57zxv6hgqql42d5r8b5k5mcq6kay-hello-2.10
```

# Description

This command deletes the store paths specified by *installables*, but only if it is safe to do so; that is, when the path is not reachable from a root of the garbage collector. This means that you can only delete paths that would also be deleted by `nix store gc`. Thus, `nix store delete` is a more targeted version of `nix store gc`.

With the option `--ignore-liveness`, reachability from the roots is ignored. However, the path still won't be deleted if there are other paths in the store that refer to it (i.e., depend on it).

# Options

- `--ignore-liveness` Do not check whether the paths are reachable from a root.

- **--stdin** Read installables from the standard input. No default installable applied.

## Common evaluation options:

- **--arg** *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- **--argstr** *name string* Pass the string *string* as the argument *name* to Nix functions.
- **--debugger** Start an interactive environment if evaluation fails.
- **--eval-store** *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- **--impure** Allow access to mutable paths and repositories.
- **--include** / **-I** *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g.

`dwarffs/nixpkgs`). This implies `--no-write-lock-file`.

- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options that change the interpretation of `installables`:

- `--all` Apply the operation to every store path.
- `--derivation` Operate on the `store derivation` rather than its outputs.

- **--expr** *expr* Interpret *installables* as attribute paths relative to the Nix expression *expr*.
- **--file** / **-f** *file* Interpret *installables* as attribute paths relative to the Nix expression stored in *file*. If *file* is the character -, then a Nix expression will be read from standard input. Implies **--impure**.
- **--recursive** / **-r** Apply operation to closure of the specified paths.

## Options to override configuration settings:

- **--accept-flake-config** Enable the `accept-flake-config` setting.
- **--access-tokens** *value* Set the `access-tokens` setting.
- **--allow-dirty** Enable the `allow-dirty` setting.
- **--allow-import-from-derivation** Enable the `allow-import-from-derivation` setting.
- **--allow-new-privileges** Enable the `allow-new-privileges` setting.
- **--allow-symlinked-store** Enable the `allow-symlinked-store` setting.
- **--allow-unsafe-native-code-during-evaluation** Enable the `allow-unsafe-native-code-during-evaluation` setting.
- **--allowed-impure-host-deps** *value* Set the `allowed-impure-host-deps` setting.
- **--allowed-uris** *value* Set the `allowed-uris` setting.
- **--allowed-users** *value* Set the `allowed-users` setting.
- **--auto-allocate-uids** Enable the `auto-allocate-uids` setting.
- **--auto-optimise-store** Enable the `auto-optimise-store` setting.
- **--bash-prompt** *value* Set the `bash-prompt` setting.
- **--bash-prompt-prefix** *value* Set the `bash-prompt-prefix` setting.
- **--bash-prompt-suffix** *value* Set the `bash-prompt-suffix` setting.
- **--build-hook** *value* Set the `build-hook` setting.
- **--build-poll-interval** *value* Set the `build-poll-interval` setting.

- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.

- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.

- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.

- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.

- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.

- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.
- `--start-id value` Set the `start-id` setting.
- `--store value` Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.

- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix store diff-closures` - show what packages and versions were added and removed between two closures

# Synopsis

`nix store diff-closures [option...] before after`

# Examples

- Show what got added and removed between two versions of the NixOS system profile:

```
nix store diff-closures /nix/var/nix/profiles/system-655-link /nix/var/nix/profiles/system-658-link
acpi-call: 2020-04-07-5.8.16 → 2020-04-07-5.8.18
baloo-widgets: 20.08.1 → 20.08.2
bluez-qt: +12.6 KiB
dolphin: 20.08.1 → 20.08.2, +13.9 KiB
kdeconnect: 20.08.2 → ∅, -6597.8 KiB
kdeconnect-kde: ∅ → 20.08.2, +6599.7 KiB
...
...
```

# Description

This command shows the differences between the two closures *before* and *after* with respect to the addition, removal, or version change of packages, as well as changes in store path sizes.

For each package name in the two closures (where a package name is defined as the name component of a store path excluding the version), if there is a change in the set of versions of the package, or a change in the size of the store paths of more than 8 KiB, it prints a line like this:

```
dolphin: 20.08.1 → 20.08.2, +13.9 KiB
```

No size change is shown if it's below the threshold. If the package does not exist in either the *before* or *after* closures, it is represented using  $\emptyset$  (empty set) on the appropriate side of the arrow. If a package has an empty version string, the version is rendered as  $\varepsilon$  (epsilon).

There may be multiple versions of a package in each closure. In that case, only the changed versions are shown. Thus,

```
libfoo: 1.2, 1.3 → 1.4
```

leaves open the possibility that there are other versions (e.g. [1.1](#)) that exist in both closures.

# Options

## Common evaluation options:

- `--arg` *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr` *name string* Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store` *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include` / `-I` *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>` ).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.

- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options that change the interpretation of `installables`:

- `--derivation` Operate on the `store derivation` rather than its outputs.
- `--expr expr` Interpret `installables` as attribute paths relative to the Nix expression *expr*.
- `--file` / `-f file` Interpret `installables` as attribute paths relative to the Nix expression stored in *file*. If *file* is the character `-`, then a Nix expression will be read from standard input. Implies `--impure`.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.

- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps`

setting.

- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.

- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.
- `--max-silent-time` *value* Set the `max-silent-time` setting.
- `--max-substitution-jobs` *value* Set the `max-substitution-jobs` setting.
- `--min-free` *value* Set the `min-free` setting.
- `--min-free-check-interval` *value* Set the `min-free-check-interval` setting.
- `--nar-buffer-size` *value* Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl` *value* Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl` *value* Set the `narinfo-cache-positive-ttl` setting.

- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.

- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.

- `--post-build-hook` *value* Set the `post-build-hook` setting.
- `--pre-build-hook` *value* Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.

- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix store dump-path` - serialise a store path to stdout in NAR format

# Synopsis

`nix store dump-path [option...] installables...`

# Examples

- To get a NAR containing the GNU Hello package:

```
nix store dump-path nixpkgs#hello > hello.nar
```

- To get a NAR from the binary cache <https://cache.nixos.org/>:

```
nix store dump-path --store https://cache.nixos.org/ \
/nix/store/7crrmih8c52r8fbnqb933dxrsp44md93-glibc-2.25 > glibc.nar
```

# Description

This command generates a NAR file containing the serialisation of the store path *installable*. The NAR is written to standard output.

# Options

- `--stdin` Read installables from the standard input. No default installable applied.

## Common evaluation options:

- `--arg` *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr` *name string* Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store` *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include` / `-I` *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the

`nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarfss/nixpkgs`). This implies `--no-write-lock-file`.

- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options that change the interpretation of `installables`:

- `--all` Apply the operation to every store path.
- `--derivation` Operate on the `store derivation` rather than its outputs.
- `--expr expr` Interpret `installables` as attribute paths relative to the Nix expression `expr`.

- **--file / -f** *file* Interpret *installables* as attribute paths relative to the Nix expression stored in *file*. If *file* is the character -, then a Nix expression will be read from standard input. Implies **--impure**.
- **--recursive / -r** Apply operation to closure of the specified paths.

## Options to override configuration settings:

- **--accept-flake-config** Enable the `accept-flake-config` setting.
- **--access-tokens** *value* Set the `access-tokens` setting.
- **--allow-dirty** Enable the `allow-dirty` setting.
- **--allow-import-from-derivation** Enable the `allow-import-from-derivation` setting.
- **--allow-new-privileges** Enable the `allow-new-privileges` setting.
- **--allow-symlinked-store** Enable the `allow-symlinked-store` setting.
- **--allow-unsafe-native-code-during-evaluation** Enable the `allow-unsafe-native-code-during-evaluation` setting.
- **--allowed-impure-host-deps** *value* Set the `allowed-impure-host-deps` setting.
- **--allowed-uris** *value* Set the `allowed-uris` setting.
- **--allowed-users** *value* Set the `allowed-users` setting.
- **--auto-allocate-uids** Enable the `auto-allocate-uids` setting.
- **--auto-optimise-store** Enable the `auto-optimise-store` setting.
- **--bash-prompt** *value* Set the `bash-prompt` setting.
- **--bash-prompt-prefix** *value* Set the `bash-prompt-prefix` setting.
- **--bash-prompt-suffix** *value* Set the `bash-prompt-suffix` setting.
- **--build-hook** *value* Set the `build-hook` setting.
- **--build-poll-interval** *value* Set the `build-poll-interval` setting.
- **--build-users-group** *value* Set the `build-users-group` setting.

- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.

- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.

- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.

- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.

- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.

- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.

- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix store gc` - perform garbage collection on a Nix store

# Synopsis

`nix store gc [option...]`

# Examples

- Delete unreachable paths in the Nix store:

```
nix store gc
```

- Delete up to 1 gigabyte of garbage:

```
nix store gc --max 1G
```

# Description

This command deletes unreachable paths in the Nix store.

# Options

- **--dry-run** Show what this command would do without doing it.
- **--max *n*** Stop after freeing *n* bytes of disk space.

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.

- `--allowed-uris value` Set the `allowed-uris` setting.
- `--allowed-users value` Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt value` Set the `bash-prompt` setting.
- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.
- `--build-hook value` Set the `build-hook` setting.
- `--build-poll-interval value` Set the `build-poll-interval` setting.
- `--build-users-group value` Set the `build-users-group` setting.
- `--builders value` Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.

- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.

- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.
- `--max-silent-time` *value* Set the `max-silent-time` setting.
- `--max-substitution-jobs` *value* Set the `max-substitution-jobs` setting.
- `--min-free` *value* Set the `min-free` setting.
- `--min-free-check-interval` *value* Set the `min-free-check-interval` setting.
- `--nar-buffer-size` *value* Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl` *value* Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl` *value* Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file` *value* Set the `netrc-file` setting.

- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.

- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.

- `--pre-build-hook` *value* Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.

- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix store ls` - show information about a path in the Nix store

## Synopsis

`nix store ls [option...] path`

# Examples

- To list the contents of a store path in a binary cache:

```
nix store ls --store https://cache.nixos.org/ --long --recursive
/nix/store/0i2jd68mp5g6h2sa5k9c85rb80sn8hi9-hello-2.10
dr-xr-xr-x 0 ./bin
-r-xr-xr-x 38184 ./bin/hello
dr-xr-xr-x 0 ./share
...
```

- To show information about a specific file in a binary cache:

```
nix store ls --store https://cache.nixos.org/ --long /nix/store
/0i2jd68mp5g6h2sa5k9c85rb80sn8hi9-hello-2.10/bin/hello
-r-xr-xr-x 38184 hello
```

# Description

This command shows information about *path* in a Nix store. *path* can be a top-level store path or any file inside a store path.

# Options

- `--directory` / `-d` Show directories rather than their contents.
- `--json` Produce output in JSON format, suitable for consumption by another program.
- `--long` / `-l` Show detailed file information.
- `--recursive` / `-R` List subdirectories recursively.

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.

- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.

- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.

- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry value` Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space value` Set the `gc-reserved-space` setting.
- `--hashed-mirrors value` Set the `hashed-mirrors` setting.
- `--http-connections value` Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count value` Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls value` Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.

- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.

- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.

- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.

- `--start-id value` Set the `start-id` setting.
- `--store value` Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix store make-content-addressed` - rewrite a path or closure to content-addressed form

# Synopsis

`nix store make-content-addressed [option...] installables...`

# Examples

- Create a content-addressed representation of the closure of GNU Hello:

```
nix store make-content-addressed nixpkgs#hello
...
rewrote '/nix/store/v5sv61sszx301i0x6xysaqzla09nksnd-hello-2.10' to
'/nix/store/5skmmcb9svys5lj3kbsrjg7vf2irid63-hello-2.10'
```

Since the resulting paths are content-addressed, they are always trusted and don't need signatures to be copied to another store:

```
nix copy --to /tmp/nix --trusted-public-keys '' /nix/store
/5skmmcb9svys5lj3kbsrjg7vf2irid63-hello-2.10
```

By contrast, the original closure is input-addressed, so it does need signatures to be trusted:

```
nix copy --to /tmp/nix --trusted-public-keys '' nixpkgs#hello
cannot add path '/nix/store/zy9wbxwcygrwnh8n2w9qbbcr6zk87m26-libunistring-
0.9.10' because it lacks a signature by a trusted key
```

- Create a content-addressed representation of the current NixOS system closure:

```
nix store make-content-addressed /run/current-system
```

# Description

This command converts the closure of the store paths specified by *installables* to content-addressed form.

Nix store paths are usually *input-addressed*, meaning that the hash part of the store path is computed from the contents of the derivation (i.e., the build-time dependency graph). Input-addressed paths need to be signed by a trusted key if you want to import them into a store, because we need to trust that the contents of the path were actually built by the derivation.

By contrast, in a *content-addressed* path, the hash part is computed from the contents of the path. This allows the contents of the path to be verified without any additional information such as signatures. This means that a command like

```
nix store build /nix/store/5skmmcb9svys5lj3kbsrjg7vf2irid63-hello-2.10 \
--substituters https://my-cache.example.org
```

will succeed even if the binary cache <https://my-cache.example.org> doesn't present any signatures.

# Options

- **--from** *store-uri* URL of the source Nix store.
- **--json** Produce output in JSON format, suitable for consumption by another program.
- **--stdin** Read installables from the standard input. No default installable applied.
- **--to** *store-uri* URL of the destination Nix store.

## Common evaluation options:

- **--arg** *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- **--argstr** *name string* Pass the string *string* as the argument *name* to Nix functions.
- **--debugger** Start an interactive environment if evaluation fails.

- `--eval-store store-url` The URL of the Nix store to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include / -I path` Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.

- `--update-input` *input-path* Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options that change the interpretation of `installables`:

- `--all` Apply the operation to every store path.
- `--derivation` Operate on the `store derivation` rather than its outputs.
- `--expr` *expr* Interpret `installables` as attribute paths relative to the Nix expression *expr*.
- `--file` / `-f` *file* Interpret `installables` as attribute paths relative to the Nix expression stored in *file*. If *file* is the character -, then a Nix expression will be read from standard input. Implies `--impure`.
- `--recursive` / `-r` Apply operation to closure of the specified paths.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.

- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors value` Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls value` Append to the `ignored-acls` setting.
- `--extra-nix-path value` Append to the `nix-path` setting.
- `--extra-platforms value` Set the `extra-platforms` setting.
- `--extra-plugin-files value` Append to the `plugin-files` setting.
- `--extra-sandbox-paths value` Append to the `sandbox-paths` setting.
- `--extra-secret-key-files value` Append to the `secret-key-files` setting.
- `--extra-substituters value` Append to the `substituters` setting.
- `--extra-system-features value` Append to the `system-features` setting.

- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.

- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.

- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.

- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.

- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.
- `--start-id value` Set the `start-id` setting.
- `--store value` Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.

- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix store optimise` - replace identical files in the store by hard links

# Synopsis

`nix store optimise [option...]`

# Examples

- Optimise the Nix store:

```
nix store optimise
```

# Description

This command deduplicates the Nix store: it scans the store for regular files with identical contents, and replaces them with hard links to a single instance.

Note that you can also set `auto-optimise-store` to `true` in `nix.conf` to perform this optimisation incrementally whenever a new path is added to the Nix store. To make this efficient, Nix maintains a content-addressed index of all the files in the Nix store in the directory `/nix/store/.links/`.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix store path-from-hash-part` - get a store path from its hash part

# Synopsis

`nix store path-from-hash-part [option...] hash-part`

# Examples

- Return the full store path with the given hash part:

```
nix store path-from-hash-part --store https://cache.nixos.org/
0i2jd68mp5g6h2sa5k9c85rb80sn8hi9
/nix/store/0i2jd68mp5g6h2sa5k9c85rb80sn8hi9-hello-2.10
```

# Description

Given the hash part of a store path (that is, the 32 characters following `/nix/store/`), return the full store path. This is primarily useful in the implementation of binary caches, where a request for a `.narinfo` file only supplies the hash part (e.g. `https://cache.nixos.org/0i2jd68mp5g6h2sa5k9c85rb80sn8hi9.narinfo`).

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix store ping` - test whether a store can be accessed

## Synopsis

`nix store ping` [*option...*]

# Examples

- Test whether connecting to a remote Nix store via SSH works:

```
nix store ping --store ssh://mac1
```

- Test whether a URL is a valid binary cache:

```
nix store ping --store https://cache.nixos.org
```

- Test whether the Nix daemon is up and running:

```
nix store ping --store daemon
```

# Description

This command tests whether a particular Nix store (specified by the argument `--store url`) can be accessed. What this means is dependent on the type of the store. For instance, for an SSH store it means that Nix can connect to the specified machine.

If the command succeeds, Nix returns a exit code of 0 and does not print any output.

# Options

- `--json` Produce output in JSON format, suitable for consumption by another program.

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.

- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps value` Set the `allowed-impure-host-deps` setting.
- `--allowed-uris value` Set the `allowed-uris` setting.
- `--allowed-users value` Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt value` Set the `bash-prompt` setting.
- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.
- `--build-hook value` Set the `build-hook` setting.
- `--build-poll-interval value` Set the `build-poll-interval` setting.
- `--build-users-group value` Set the `build-users-group` setting.
- `--builders value` Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.

- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.

- `--gc-reserved-space value` Set the `gc-reserved-space` setting.
- `--hashed-mirrors value` Set the `hashed-mirrors` setting.
- `--http-connections value` Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count value` Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls value` Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.

- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.

- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.

- `--plugin-files` *value* Set the `plugin-files` setting.
- `--post-build-hook` *value* Set the `post-build-hook` setting.
- `--pre-build-hook` *value* Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.

- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix store prefetch-file` - download a file into the Nix store

## Synopsis

`nix store prefetch-file [option...] url`

# Examples

- Download a file to the Nix store:

```
nix store prefetch-file https://releases.nixos.org/nix/nix-2.3.10
/nix-2.3.10.tar.xz
Downloaded 'https://releases.nixos.org/nix/nix-2.3.10/nix-2.3.10.tar.xz' to
'/nix/store/vbdbi42hgnc4h7pyqzp6h2yf77kw93aw-source' (hash
'sha256-qKheVd5D0BervxMDbt+1hnTKE2aRWC8XCAwc0SeHt6s=').
```

- Download a file and get the SHA-512 hash:

```
nix store prefetch-file --json --hash-type sha512 \
https://releases.nixos.org/nix/nix-2.3.10/nix-2.3.10.tar.xz \
| jq -r .hash
sha512-6XJxfym0TNH9knxeH4Z0vns6wElFy3uahunl2hJgovACCMEMXSy42s69zWVyGJALXTI+
86tpDJGLIcAySEKBbA==
```

# Description

This command downloads the file *url* to the Nix store. It prints out the resulting store path

and the cryptographic hash of the contents of the file.

The name component of the store path defaults to the last component of *url*, but this can be overridden using `--name`.

# Options

- `--executable` Make the resulting file executable. Note that this causes the resulting hash to be a NAR hash rather than a flat file hash.
- `--expected-hash` *hash* The expected hash of the file.
- `--hash-type` *hash-algo* hash algorithm ('md5', 'sha1', 'sha256', or 'sha512')
- `--json` Produce output in JSON format, suitable for consumption by another program.
- `--name` *name* Override the name component of the resulting store path. It defaults to the base name of *url*.

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.

- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.

- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.
- `--extra-extra-platforms value` Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors value` Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls value` Append to the `ignored-acls` setting.
- `--extra-nix-path value` Append to the `nix-path` setting.
- `--extra-platforms value` Set the `extra-platforms` setting.
- `--extra-plugin-files value` Append to the `plugin-files` setting.
- `--extra-sandbox-paths value` Append to the `sandbox-paths` setting.
- `--extra-secret-key-files value` Append to the `secret-key-files` setting.
- `--extra-substituters value` Append to the `substituters` setting.

- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.

- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.

- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.

- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.

- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.
- `--start-id value` Set the `start-id` setting.
- `--store value` Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.

- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix store repair` - repair store paths

## Synopsis

`nix store repair [option...] installables...`

# Examples

- Repair a store path, after determining that it is corrupt:

```
nix store verify /nix/store/yb5q57z xv6hgqqql42d5r8b5k5mcq6kay-hello-2.10
path '/nix/store/yb5q57z xv6hgqqql42d5r8b5k5mcq6kay-hello-2.10' was
modified! expected hash
'sha256:1hd5vnh6xjk388gdk841vflicy8qv7qzj2hb7xlyh8lpb43j921l', got
'sha256:1a25lf78x5wi6pfkrxalf0n13kdaca0bqmjqnp7wfjza2qz5ssgl'

nix store repair /nix/store/yb5q57z xv6hgqqql42d5r8b5k5mcq6kay-hello-2.10
```

# Description

This command attempts to "repair" the store paths specified by *installables* by redownloading them using the available substituters. If no substitutes are available, then repair is not possible.

---

## Warning

During repair, there is a very small time window during which the old path (if it exists)

is moved out of the way and replaced with the new path. If repair is interrupted in between, then the system may be left in a broken state (e.g., if the path contains a critical system component like the GNU C Library).

---

# Options

- `--stdin` Read installables from the standard input. No default installable applied.

## Common evaluation options:

- `--arg` *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr` *name string* Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store` *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include` / `-I` *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>` ).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.

- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file` *flake-lock-path* Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input` *input-path flake-url* Override a specific flake input (e.g. `dwarfss/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file` *flake-lock-path* Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input` *input-path* Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format` *format* Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.

- **--repair** During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- **--version** Show version information.

## Options that change the interpretation of `installables`:

- **--all** Apply the operation to every store path.
- **--derivation** Operate on the `store derivation` rather than its outputs.
- **--expr** *expr* Interpret `installables` as attribute paths relative to the Nix expression *expr*.
- **--file** / **-f** *file* Interpret `installables` as attribute paths relative to the Nix expression stored in *file*. If *file* is the character -, then a Nix expression will be read from standard input. Implies **--impure**.
- **--recursive** / **-r** Apply operation to closure of the specified paths.

## Options to override configuration settings:

- **--accept-flake-config** Enable the `accept-flake-config` setting.
- **--access-tokens** *value* Set the `access-tokens` setting.
- **--allow-dirty** Enable the `allow-dirty` setting.
- **--allow-import-from-derivation** Enable the `allow-import-from-derivation` setting.
- **--allow-new-privileges** Enable the `allow-new-privileges` setting.
- **--allow-symlinked-store** Enable the `allow-symlinked-store` setting.
- **--allow-unsafe-native-code-during-evaluation** Enable the `allow-unsafe-native-code-during-evaluation` setting.
- **--allowed-impure-host-deps** *value* Set the `allowed-impure-host-deps` setting.
- **--allowed-uris** *value* Set the `allowed-uris` setting.
- **--allowed-users** *value* Set the `allowed-users` setting.
- **--auto-allocate-uids** Enable the `auto-allocate-uids` setting.
- **--auto-optimise-store** Enable the `auto-optimise-store` setting.

- `--bash-prompt value` Set the `bash-prompt` setting.
- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.
- `--build-hook value` Set the `build-hook` setting.
- `--build-poll-interval value` Set the `build-poll-interval` setting.
- `--build-users-group value` Set the `build-users-group` setting.
- `--builders value` Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features value` Set the `experimental-features` setting.
- `--extra-access-tokens value` Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps value` Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris value` Append to the `allowed-uris` setting.
- `--extra-allowed-users value` Append to the `allowed-users` setting.
- `--extra-build-hook value` Append to the `build-hook` setting.
- `--extra-experimental-features value` Append to the `experimental-features` setting.

- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.

- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.

- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.

- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.

- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir` *value* Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size` *value* Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths` *value* Set the `sandbox-paths` setting.
- `--secret-key-files` *value* Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file` *value* Set the `ssl-cert-file` setting.
- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.

- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix store sign` - sign store paths

# Synopsis

`nix store sign` [*option...*] *installables...*

# Options

- `--key-file` / `-k` *file* File containing the secret signing key.
- `--stdin` Read installables from the standard input. No default installable applied.

## Common evaluation options:

- `--arg` *name expr* Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr` *name string* Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store` *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include` / `-I` *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the location of Nix expressions using `paths` enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for `nix-channel`](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

## Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.

- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option` *name value* Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options that change the interpretation of `installables`:

- `--all` Apply the operation to every store path.
- `--derivation` Operate on the `store derivation` rather than its outputs.
- `--expr` *expr* Interpret `installables` as attribute paths relative to the Nix expression *expr*.
- `--file` / `-f` *file* Interpret `installables` as attribute paths relative to the Nix expression stored in *file*. If *file* is the character `-`, then a Nix expression will be read from standard input. Implies `--impure`.
- `--recursive` / `-r` Apply operation to closure of the specified paths.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens` *value* Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.

- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps value` Set the `allowed-impure-host-deps` setting.
- `--allowed-uris value` Set the `allowed-uris` setting.
- `--allowed-users value` Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt value` Set the `bash-prompt` setting.
- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.
- `--build-hook value` Set the `build-hook` setting.
- `--build-poll-interval value` Set the `build-poll-interval` setting.
- `--build-users-group value` Set the `build-users-group` setting.
- `--builders value` Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.

- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.

- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space value` Set the `gc-reserved-space` setting.
- `--hashed-mirrors value` Set the `hashed-mirrors` setting.
- `--http-connections value` Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count value` Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls value` Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.

- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.

- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.

- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.
- `--start-id value` Set the `start-id` setting.
- `--store value` Set the `store` setting.

- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix store verify` - verify the integrity of store paths

# Synopsis

`nix store verify [option...] installables...`

# Examples

- Verify the entire Nix store:

```
nix store verify --all
```

- Check whether each path in the closure of Firefox has at least 2 signatures:

```
nix store verify --recursive --sigs-needed 2 --no-contents $(type -p firefox)
```

- Verify a store path in the binary cache <https://cache.nixos.org/> :

```
nix store verify --store https://cache.nixos.org/ \
/nix/store/v5sv61sszx301i0x6xysaqzla09nksnd-hello-2.10
```

# Description

This command verifies the integrity of the store paths *installables*, or, if `--all` is given, the entire Nix store. For each path, it checks that

- its contents match the NAR hash recorded in the Nix database; and
- it is *trusted*, that is, it is signed by at least one trusted signing key, is content-addressed, or is built locally ("ultimately trusted").

## Exit status

The exit status of this command is the sum of the following values:

- **1** if any path is corrupted (i.e. its contents don't match the recorded NAR hash).
- **2** if any path is untrusted.
- **4** if any path couldn't be verified for any other reason (such as an I/O error).

## Options

- **--no-contents** Do not verify the contents of each store path.
- **--no-trust** Do not verify whether each store path is trusted.
- **--sigs-needed / -n *n*** Require that each path is signed by at least *n* different keys.
- **--stdin** Read installables from the standard input. No default installable applied.
- **--substituter / -s *store-uri*** Use signatures from the specified store.

### Common evaluation options:

- **--arg *name expr*** Pass the value *expr* as the argument *name* to Nix functions.
- **--argstr *name string*** Pass the string *string* as the argument *name* to Nix functions.
- **--debugger** Start an interactive environment if evaluation fails.
- **--eval-store *store-url*** The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations ([.drv](#) files) and inputs referenced by them.
- **--impure** Allow access to mutable paths and repositories.
- **--include / -I *path*** Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated **NIX\_PATH** environment variable, and is used to look up the

location of Nix expressions using [paths](#) enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for nix-channel](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

### Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

### Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.

- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options that change the interpretation of `installables`:

- `--all` Apply the operation to every store path.
- `--derivation` Operate on the `store derivation` rather than its outputs.
- `--expr expr` Interpret `installables` as attribute paths relative to the Nix expression *expr*.
- `--file` / `-f file` Interpret `installables` as attribute paths relative to the Nix expression stored in *file*. If *file* is the character -, then a Nix expression will be read from standard input. Implies `--impure`.
- `--recursive` / `-r` Apply operation to closure of the specified paths.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation`

setting.

- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.
- `--download-attempts` *value* Set the `download-attempts` setting.

- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.

- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry value` Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space value` Set the `gc-reserved-space` setting.
- `--hashed-mirrors value` Set the `hashed-mirrors` setting.
- `--http-connections value` Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count value` Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls value` Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.

- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.

- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.

- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.

- `--start-id value` Set the `start-id` setting.
- `--store value` Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix upgrade-nix` - upgrade Nix to the stable version declared in Nixpkgs

# Synopsis

`nix upgrade-nix [option...]`

# Examples

- Upgrade Nix to the stable version declared in Nixpkgs:

```
nix upgrade-nix
```

- Upgrade Nix in a specific profile:

```
nix upgrade-nix --profile ~alice/.local/state/nix/profiles/profile
```

# Description

This command upgrades Nix to the stable version declared in Nixpkgs. This stable version is defined in `nix-fallback-paths.nix` and updated manually. It may not always be the latest tagged release.

By default, it locates the directory containing the `nix` binary in the `$PATH` environment variable. If that directory is a Nix profile, it will upgrade the `nix` package in that profile to the latest stable binary release.

You cannot use this command to upgrade Nix in the system profile of a NixOS system (that

is, if `nix` is found in `/run/current-system`).

# Options

- `--dry-run` Show what this command would do without doing it.
- `--nix-store-paths-url url` The URL of the file that contains the store paths of the latest Nix release.
- `--profile / -p profile-dir` The path to the Nix profile to upgrade.

## Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.
- `--print-build-logs / -L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose / -v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting `name` to `value` (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--version` Show version information.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.

- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.
- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps` *value* Set the `allowed-impure-host-deps` setting.
- `--allowed-uris` *value* Set the `allowed-uris` setting.
- `--allowed-users` *value* Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt` *value* Set the `bash-prompt` setting.
- `--bash-prompt-prefix` *value* Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix` *value* Set the `bash-prompt-suffix` setting.
- `--build-hook` *value* Set the `build-hook` setting.
- `--build-poll-interval` *value* Set the `build-poll-interval` setting.
- `--build-users-group` *value* Set the `build-users-group` setting.
- `--builders` *value* Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary` *value* Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout` *value* Set the `connect-timeout` setting.
- `--cores` *value* Set the `cores` setting.
- `--diff-hook` *value* Set the `diff-hook` setting.

- `--download-attempts` *value* Set the `download-attempts` setting.
- `--download-speed` *value* Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.
- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.

- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.
- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space` *value* Set the `gc-reserved-space` setting.
- `--hashed-mirrors` *value* Set the `hashed-mirrors` setting.
- `--http-connections` *value* Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count` *value* Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls` *value* Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines` *value* Set the `log-lines` setting.
- `--max-build-log-size` *value* Set the `max-build-log-size` setting.
- `--max-free` *value* Set the `max-free` setting.
- `--max-jobs` *value* Set the `max-jobs` setting.
- `--max-silent-time` *value* Set the `max-silent-time` setting.
- `--max-substitution-jobs` *value* Set the `max-substitution-jobs` setting.

- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.
- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.

- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.
- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.

- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.
- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.

- `--stalled-download-timeout` *value* Set the `stalled-download-timeout` setting.
- `--start-id` *value* Set the `start-id` setting.
- `--store` *value* Set the `store` setting.
- `--substitute` Enable the `substitute` setting.
- `--substituters` *value* Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system` *value* Set the `system` setting.
- `--system-features` *value* Set the `system-features` setting.
- `--tarball-ttl` *value* Set the `tarball-ttl` setting.
- `--timeout` *value* Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys` *value* Set the `trusted-public-keys` setting.
- `--trusted-substituters` *value* Set the `trusted-substituters` setting.
- `--trusted-users` *value* Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix` *value* Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

---

## Warning

This program is **experimental** and its interface is subject to change.

---

# Name

`nix why-depends` - show why a package has another package in its closure

# Synopsis

`nix why-depends [option...] package dependency`

# Examples

- Show one path through the dependency graph leading from Hello to Glibc:

```
nix why-depends nixpkgs#hello nixpkgs#glibc
/nix/store/v5sv61sszx301i0x6xysaqzla09nksnd-hello-2.10
└─bin/hello:/nix/store
/9l06v7fc38c1x3r2iydl15ksgz0ysb82-glibc-2.32/lib/ld-linux-x86-64...
 → /nix/store/9l06v7fc38c1x3r2iydl15ksgz0ysb82-glibc-2.32
```

- Show all files and paths in the dependency graph leading from Thunderbird to libX11:

```
nix why-depends --all nixpkgs#thunderbird nixpkgs#xorg.libX11
/nix/store/qfc8729nzpdln1h0hvi1zicls13m84sr-thunderbird-78.5.1
└─lib/thunderbird/libxul.so: ...6wrw-libxcb-1.14/lib:/nix/store
/adzfjjh8w25vdr0xdx9x16ah4f5rqrw5-libX11-1.7.0/lib:/nix/store/ssf...
| → /nix/store/adzfjjh8w25vdr0xdx9x16ah4f5rqrw5-libX11-1.7.0
└─lib/thunderbird/libxul.so: ...pxyc-libXt-1.2.0/lib:/nix/store
/1qj29ipxl2fyi2b13l39hdircq17gnk0-libXdamage-1.1.5/lib:/nix/store...
| → /nix/store/1qj29ipxl2fyi2b13l39hdircq17gnk0-libXdamage-1.1.5
| └─lib/libXdamage.so.1.1.0: ...libXfixes-5.0.3/lib:/nix/store
/adzfjjh8w25vdr0xdx9x16ah4f5rqrw5-libX11-1.7.0/lib:/nix/store/9l0...
| | → /nix/store/adzfjjh8w25vdr0xdx9x16ah4f5rqrw5-libX11-1.7.0
...
...
```

- Show why Glibc depends on itself:

```
nix why-depends nixpkgs#glibc nixpkgs#glibc
/nix/store/9df65igwjmfp2bw0gbrrgair6piqjgmi-glibc-2.31
└─lib/ld-2.31.so: ...che Do not use /nix/store
/9df65igwjmfp2bw0gbrrgair6piqjgmi-glibc-2.31/etc/ld.so.cache. --...
→ /nix/store/9df65igwjmfp2bw0gbrrgair6piqjgmi-glibc-2.31
```

- Show why Geeqie has a build-time dependency on `systemd`:

```
nix why-depends --derivation nixpkgs#geeqie nixpkgs#systemd
/nix/store/drrpq2fqqlrbj98bmazrnww7hm1in3wgj-geeqie-1.4.drv
└─/: ...atch.drv", ["out"]), ("/nix/store/qzh8dyq3lfbk3i1acbp7x9wh3il2imiv-
gtk+3-3.24.21.drv", ["dev"]), ("...
→ /nix/store/qzh8dyq3lfbk3i1acbp7x9wh3il2imiv-gtk+3-3.24.21.drv
└─/: ...16.0.drv", ["dev"]), ("/nix/store
/8kp79fyslf3z4m3dpvlh6w46iaadz5c2-cups-2.3.3.drv", ["dev"]), ("/nix...
→ /nix/store/8kp79fyslf3z4m3dpvlh6w46iaadz5c2-cups-2.3.3.drv
└─/:3.1.drv", ["out"]), ("/nix/store
/yd3ihapyi5wbz1kjacq9dbkaq5v5hqjg-systemd-246.4.drv", ["dev"]), ("...
→ /nix/store/yd3ihapyi5wbz1kjacq9dbkaq5v5hqjg-systemd-246.4.drv
```

# Description

Nix automatically determines potential runtime dependencies between store paths by

scanning for the *hash parts* of store paths. For instance, if there exists a store path `/nix/store/9df65igwjm...f2wbw0gbrrgair6piqjgmi-glibc-2.31`, and a file inside another store path contains the string `9df65igw...`, then the latter store path *refers* to the former, and thus might need it at runtime. Nix always maintains the existence of the transitive closure of a store path under the references relationship; it is therefore not possible to install a store path without having all of its references present.

Sometimes Nix packages end up with unexpected runtime dependencies; for instance, a reference to a compiler might accidentally end up in a binary, causing the former to be in the latter's closure. This kind of *closure size bloat* is undesirable.

`nix why-depends` allows you to diagnose the cause of such issues. It shows why the store path *package* depends on the store path *dependency*, by showing a shortest sequence in the references graph from the former to the latter. Also, for each node along this path, it shows a file fragment containing a reference to the next store path in the sequence.

To show why derivation *package* has a build-time rather than runtime dependency on derivation *dependency*, use `--derivation`.

# Options

- `--all` / `-a` Show all edges in the dependency graph leading from *package* to *dependency*, rather than just a shortest path.
- `--precise` For each edge in the dependency graph, show the files in the parent that cause the dependency.

## Common evaluation options:

- `--arg` *name* *expr* Pass the value *expr* as the argument *name* to Nix functions.
- `--argstr` *name* *string* Pass the string *string* as the argument *name* to Nix functions.
- `--debugger` Start an interactive environment if evaluation fails.
- `--eval-store` *store-url* The [URL of the Nix store](#) to use for evaluation, i.e. to store derivations (`.drv` files) and inputs referenced by them.
- `--impure` Allow access to mutable paths and repositories.
- `--include` / `-I` *path* Add *path* to the Nix search path. The Nix search path is initialized from the colon-separated `NIX_PATH` environment variable, and is used to look up the

location of Nix expressions using [paths](#) enclosed in angle brackets (i.e., `<nixpkgs>`).

For instance, passing

```
-I /home/eelco/Dev
-I /etc/nixos
```

will cause Nix to look for paths relative to `/home/eelco/Dev` and `/etc/nixos`, in that order. This is equivalent to setting the `NIX_PATH` environment variable to

```
/home/eelco/Dev:/etc/nixos
```

It is also possible to match paths against a prefix. For example, passing

```
-I nixpkgs=/home/eelco/Dev/nixpkgs-branch
-I /etc/nixos
```

will cause Nix to search for `<nixpkgs/path>` in `/home/eelco/Dev/nixpkgs-branch/path` and `/etc/nixos/nixpkgs/path`.

If a path in the Nix search path starts with `http://` or `https://`, it is interpreted as the URL of a tarball that will be downloaded and unpacked to a temporary location. The tarball must consist of a single top-level directory. For example, passing

```
-I nixpkgs=https://github.com/NixOS/nixpkgs/archive/master.tar.gz
```

tells Nix to download and use the current contents of the `master` branch in the `nixpkgs` repository.

The URLs of the tarballs from the official `nixos.org` channels (see [the manual page for nix-channel](#)) can be abbreviated as `channel:<channel-name>`. For instance, the following two flags are equivalent:

```
-I nixpkgs=channel:nixos-21.05
-I nixpkgs=https://nixos.org/channels/nixos-21.05/nixexprs.tar.xz
```

You can also fetch source trees using [flake URLs](#) and add them to the search path. For instance,

```
-I nixpkgs=flake:nixpkgs
```

specifies that the prefix `nixpkgs` shall refer to the source tree downloaded from the `nixpkgs` entry in the flake registry. Similarly,

```
-I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05
```

makes `<nixpkgs>` refer to a particular branch of the `NixOS/nixpkgs` repository on GitHub.

- `--override-flake original-ref resolved-ref` Override the flake registries, redirecting `original-ref` to `resolved-ref`.

### Common flake-related options:

- `--commit-lock-file` Commit changes to the flake's lock file.
- `--inputs-from flake-url` Use the inputs of the specified flake as registry entries.
- `--no-registries` Don't allow lookups in the flake registries. This option is deprecated; use `--no-use-registries`.
- `--no-update-lock-file` Do not allow any updates to the flake's lock file.
- `--no-write-lock-file` Do not write the flake's newly generated lock file.
- `--output-lock-file flake-lock-path` Write the given lock file instead of `flake.lock` within the top-level flake.
- `--override-input input-path flake-url` Override a specific flake input (e.g. `dwarffs/nixpkgs`). This implies `--no-write-lock-file`.
- `--recreate-lock-file` Recreate the flake's lock file from scratch.
- `--reference-lock-file flake-lock-path` Read the given lock file instead of `flake.lock` within the top-level flake.
- `--update-input input-path` Update a specific flake input (ignoring its previous entry in the lock file).

### Logging-related options:

- `--debug` Set the logging verbosity level to 'debug'.
- `--log-format format` Set the format of log output; one of `raw`, `internal-json`, `bar` or `bar-with-logs`.

- `--print-build-logs` / `-L` Print full build logs on standard error.
- `--quiet` Decrease the logging verbosity level.
- `--verbose` / `-v` Increase the logging verbosity level.

## Miscellaneous global options:

- `--help` Show usage information.
- `--offline` Disable substituters and consider all previously downloaded files up-to-date.
- `--option name value` Set the Nix configuration setting *name* to *value* (overriding `nix.conf`).
- `--refresh` Consider all previously downloaded files out-of-date.
- `--repair` During evaluation, rewrite missing or corrupted files in the Nix store. During building, rebuild missing or corrupted store paths.
- `--version` Show version information.

## Options that change the interpretation of `installables`:

- `--derivation` Operate on the `store derivation` rather than its outputs.
- `--expr expr` Interpret `installables` as attribute paths relative to the Nix expression *expr*.
- `--file` / `-f file` Interpret `installables` as attribute paths relative to the Nix expression stored in *file*. If *file* is the character `-`, then a Nix expression will be read from standard input. Implies `--impure`.

## Options to override configuration settings:

- `--accept-flake-config` Enable the `accept-flake-config` setting.
- `--access-tokens value` Set the `access-tokens` setting.
- `--allow-dirty` Enable the `allow-dirty` setting.
- `--allow-import-from-derivation` Enable the `allow-import-from-derivation` setting.
- `--allow-new-privileges` Enable the `allow-new-privileges` setting.

- `--allow-symlinked-store` Enable the `allow-symlinked-store` setting.
- `--allow-unsafe-native-code-during-evaluation` Enable the `allow-unsafe-native-code-during-evaluation` setting.
- `--allowed-impure-host-deps value` Set the `allowed-impure-host-deps` setting.
- `--allowed-uris value` Set the `allowed-uris` setting.
- `--allowed-users value` Set the `allowed-users` setting.
- `--auto-allocate-uids` Enable the `auto-allocate-uids` setting.
- `--auto-optimise-store` Enable the `auto-optimise-store` setting.
- `--bash-prompt value` Set the `bash-prompt` setting.
- `--bash-prompt-prefix value` Set the `bash-prompt-prefix` setting.
- `--bash-prompt-suffix value` Set the `bash-prompt-suffix` setting.
- `--build-hook value` Set the `build-hook` setting.
- `--build-poll-interval value` Set the `build-poll-interval` setting.
- `--build-users-group value` Set the `build-users-group` setting.
- `--builders value` Set the `builders` setting.
- `--builders-use-substitutes` Enable the `builders-use-substitutes` setting.
- `--commit-lockfile-summary value` Set the `commit-lockfile-summary` setting.
- `--compress-build-log` Enable the `compress-build-log` setting.
- `--connect-timeout value` Set the `connect-timeout` setting.
- `--cores value` Set the `cores` setting.
- `--diff-hook value` Set the `diff-hook` setting.
- `--download-attempts value` Set the `download-attempts` setting.
- `--download-speed value` Set the `download-speed` setting.
- `--eval-cache` Enable the `eval-cache` setting.

- `--experimental-features` *value* Set the `experimental-features` setting.
- `--extra-access-tokens` *value* Append to the `access-tokens` setting.
- `--extra-allowed-impure-host-deps` *value* Append to the `allowed-impure-host-deps` setting.
- `--extra-allowed-uris` *value* Append to the `allowed-uris` setting.
- `--extra-allowed-users` *value* Append to the `allowed-users` setting.
- `--extra-build-hook` *value* Append to the `build-hook` setting.
- `--extra-experimental-features` *value* Append to the `experimental-features` setting.
- `--extra-extra-platforms` *value* Append to the `extra-platforms` setting.
- `--extra-hashed-mirrors` *value* Append to the `hashed-mirrors` setting.
- `--extra-ignored-acls` *value* Append to the `ignored-acls` setting.
- `--extra-nix-path` *value* Append to the `nix-path` setting.
- `--extra-platforms` *value* Set the `extra-platforms` setting.
- `--extra-plugin-files` *value* Append to the `plugin-files` setting.
- `--extra-sandbox-paths` *value* Append to the `sandbox-paths` setting.
- `--extra-secret-key-files` *value* Append to the `secret-key-files` setting.
- `--extra-substituters` *value* Append to the `substituters` setting.
- `--extra-system-features` *value* Append to the `system-features` setting.
- `--extra-trusted-public-keys` *value* Append to the `trusted-public-keys` setting.
- `--extra-trusted-substituters` *value* Append to the `trusted-substituters` setting.
- `--extra-trusted-users` *value* Append to the `trusted-users` setting.
- `--fallback` Enable the `fallback` setting.
- `--filter-syscalls` Enable the `filter-syscalls` setting.
- `--flake-registry` *value* Set the `flake-registry` setting.

- `--fsync-metadata` Enable the `fsync-metadata` setting.
- `--gc-reserved-space value` Set the `gc-reserved-space` setting.
- `--hashed-mirrors value` Set the `hashed-mirrors` setting.
- `--http-connections value` Set the `http-connections` setting.
- `--http2` Enable the `http2` setting.
- `--id-count value` Set the `id-count` setting.
- `--ignore-try` Enable the `ignore-try` setting.
- `--ignored-acls value` Set the `ignored-acls` setting.
- `--impersonate-linux-26` Enable the `impersonate-linux-26` setting.
- `--keep-build-log` Enable the `keep-build-log` setting.
- `--keep-derivations` Enable the `keep-derivations` setting.
- `--keep-env-derivations` Enable the `keep-env-derivations` setting.
- `--keep-failed` Enable the `keep-failed` setting.
- `--keep-going` Enable the `keep-going` setting.
- `--keep-outputs` Enable the `keep-outputs` setting.
- `--log-lines value` Set the `log-lines` setting.
- `--max-build-log-size value` Set the `max-build-log-size` setting.
- `--max-free value` Set the `max-free` setting.
- `--max-jobs value` Set the `max-jobs` setting.
- `--max-silent-time value` Set the `max-silent-time` setting.
- `--max-substitution-jobs value` Set the `max-substitution-jobs` setting.
- `--min-free value` Set the `min-free` setting.
- `--min-free-check-interval value` Set the `min-free-check-interval` setting.
- `--nar-buffer-size value` Set the `nar-buffer-size` setting.

- `--narinfo-cache-negative-ttl value` Set the `narinfo-cache-negative-ttl` setting.
- `--narinfo-cache-positive-ttl value` Set the `narinfo-cache-positive-ttl` setting.
- `--netrc-file value` Set the `netrc-file` setting.
- `--nix-path value` Set the `nix-path` setting.
- `--no-accept-flake-config` Disable the `accept-flake-config` setting.
- `--no-allow-dirty` Disable the `allow-dirty` setting.
- `--no-allow-import-from-derivation` Disable the `allow-import-from-derivation` setting.
- `--no-allow-new-privileges` Disable the `allow-new-privileges` setting.
- `--no-allow-symlinked-store` Disable the `allow-symlinked-store` setting.
- `--no-allow-unsafe-native-code-during-evaluation` Disable the `allow-unsafe-native-code-during-evaluation` setting.
- `--no-auto-allocate-uids` Disable the `auto-allocate-uids` setting.
- `--no-auto-optimise-store` Disable the `auto-optimise-store` setting.
- `--no-builders-use-substitutes` Disable the `builders-use-substitutes` setting.
- `--no-compress-build-log` Disable the `compress-build-log` setting.
- `--no-eval-cache` Disable the `eval-cache` setting.
- `--no-fallback` Disable the `fallback` setting.
- `--no-filter-syscalls` Disable the `filter-syscalls` setting.
- `--no-fsync-metadata` Disable the `fsync-metadata` setting.
- `--no-http2` Disable the `http2` setting.
- `--no-ignore-try` Disable the `ignore-try` setting.
- `--no-impersonate-linux-26` Disable the `impersonate-linux-26` setting.
- `--no-keep-build-log` Disable the `keep-build-log` setting.
- `--no-keep-derivations` Disable the `keep-derivations` setting.

- `--no-keep-env-derivations` Disable the `keep-env-derivations` setting.
- `--no-keep-failed` Disable the `keep-failed` setting.
- `--no-keep-going` Disable the `keep-going` setting.
- `--no-keep-outputs` Disable the `keep-outputs` setting.
- `--no-preallocate-contents` Disable the `preallocate-contents` setting.
- `--no-print-missing` Disable the `print-missing` setting.
- `--no-pure-eval` Disable the `pure-eval` setting.
- `--no-require-drop-supplementary-groups` Disable the `require-drop-supplementary-groups` setting.
- `--no-require-sigs` Disable the `require-sigs` setting.
- `--no-restrict-eval` Disable the `restrict-eval` setting.
- `--no-run-diff-hook` Disable the `run-diff-hook` setting.
- `--no-sandbox` Disable sandboxing.
- `--no-sandbox-fallback` Disable the `sandbox-fallback` setting.
- `--no-show-trace` Disable the `show-trace` setting.
- `--no-substitute` Disable the `substitute` setting.
- `--no-sync-before-registering` Disable the `sync-before-registering` setting.
- `--no-trace-function-calls` Disable the `trace-function-calls` setting.
- `--no-trace-verbose` Disable the `trace-verbose` setting.
- `--no-use-case-hack` Disable the `use-case-hack` setting.
- `--no-use-cgroups` Disable the `use-cgroups` setting.
- `--no-use-registries` Disable the `use-registries` setting.
- `--no-use-sqlite-wal` Disable the `use-sqlite-wal` setting.
- `--no-use-xdg-base-directories` Disable the `use-xdg-base-directories` setting.

- `--no-warn-dirty` Disable the `warn-dirty` setting.
- `--plugin-files value` Set the `plugin-files` setting.
- `--post-build-hook value` Set the `post-build-hook` setting.
- `--pre-build-hook value` Set the `pre-build-hook` setting.
- `--preallocate-contents` Enable the `preallocate-contents` setting.
- `--print-missing` Enable the `print-missing` setting.
- `--pure-eval` Enable the `pure-eval` setting.
- `--relaxed-sandbox` Enable sandboxing, but allow builds to disable it.
- `--require-drop-supplementary-groups` Enable the `require-drop-supplementary-groups` setting.
- `--require-sigs` Enable the `require-sigs` setting.
- `--restrict-eval` Enable the `restrict-eval` setting.
- `--run-diff-hook` Enable the `run-diff-hook` setting.
- `--sandbox` Enable sandboxing.
- `--sandbox-build-dir value` Set the `sandbox-build-dir` setting.
- `--sandbox-dev-shm-size value` Set the `sandbox-dev-shm-size` setting.
- `--sandbox-fallback` Enable the `sandbox-fallback` setting.
- `--sandbox-paths value` Set the `sandbox-paths` setting.
- `--secret-key-files value` Set the `secret-key-files` setting.
- `--show-trace` Enable the `show-trace` setting.
- `--ssl-cert-file value` Set the `ssl-cert-file` setting.
- `--stalled-download-timeout value` Set the `stalled-download-timeout` setting.
- `--start-id value` Set the `start-id` setting.
- `--store value` Set the `store` setting.

- `--substitute` Enable the `substitute` setting.
- `--substituters value` Set the `substituters` setting.
- `--sync-before-registering` Enable the `sync-before-registering` setting.
- `--system value` Set the `system` setting.
- `--system-features value` Set the `system-features` setting.
- `--tarball-ttl value` Set the `tarball-ttl` setting.
- `--timeout value` Set the `timeout` setting.
- `--trace-function-calls` Enable the `trace-function-calls` setting.
- `--trace-verbose` Enable the `trace-verbose` setting.
- `--trusted-public-keys value` Set the `trusted-public-keys` setting.
- `--trusted-substituters value` Set the `trusted-substituters` setting.
- `--trusted-users value` Set the `trusted-users` setting.
- `--use-case-hack` Enable the `use-case-hack` setting.
- `--use-cgroups` Enable the `use-cgroups` setting.
- `--use-registries` Enable the `use-registries` setting.
- `--use-sqlite-wal` Enable the `use-sqlite-wal` setting.
- `--use-xdg-base-directories` Enable the `use-xdg-base-directories` setting.
- `--user-agent-suffix value` Set the `user-agent-suffix` setting.
- `--warn-dirty` Enable the `warn-dirty` setting.

# Files

This section lists configuration files that you can use when you work with Nix.

# Name

`nix.conf` - Nix configuration file

# Description

Nix supports a variety of configuration settings, which are read from configuration files or taken as command line flags.

## Configuration file

By default Nix reads settings from the following places, in that order:

1. The system-wide configuration file `sysconfdir/nix/nix.conf` (i.e. `/etc/nix/nix.conf` on most systems), or `$NIX_CONF_DIR/nix.conf` if `NIX_CONF_DIR` is set.

Values loaded in this file are not forwarded to the Nix daemon. The client assumes that the daemon has already loaded them.

2. If `NIX_USER_CONF_FILES` is set, then each path separated by `:` will be loaded in reverse order.

Otherwise it will look for `nix/nix.conf` files in `XDG_CONFIG_DIRS` and `XDG_CONFIG_HOME`. If unset, `XDG_CONFIG_DIRS` defaults to `/etc/xdg`, and `XDG_CONFIG_HOME` defaults to `$HOME/.config` as per [XDG Base Directory Specification](#).

3. If `NIX_CONFIG` is set, its contents are treated as the contents of a configuration file.

## File format

Configuration files consist of `name = value` pairs, one per line. Comments start with a `#` character.

Example:

```
keep-outputs = true # Nice for developers
keep-derivations = true # Idem
```

Other files can be included with a line like `include <path>`, where `<path>` is interpreted

relative to the current configuration file. A missing file is an error unless `!include` is used instead.

A configuration setting usually overrides any previous value. However, for settings that take a list of items, you can prefix the name of the setting by `extra-` to *append* to the previous value.

For instance,

```
substituters = a b
extra-substituters = c d
```

defines the `substituters` setting to be `a b c d`.

Unknown option names are not an error, and are simply ignored with a warning.

## Command line flags

Configuration options can be set on the command line, overriding the values set in the [configuration file](#):

- Every configuration setting has corresponding command line flag (e.g. `--max-jobs 16`). Boolean settings do not need an argument, and can be explicitly disabled with the `no-` prefix (e.g. `--keep-failed` and `--no-keep-failed`).

Unknown option names are invalid flags (unless there is already a flag with that name), and are rejected with an error.

- The flag `--option <name> <value>` is interpreted exactly like a `<name> = <value>` in a setting file.

Unknown option names are ignored with a warning.

The `extra-` prefix is supported for settings that take a list of items (e.g. `--extra-trusted-users alice` or `--option extra-trusted-users alice`).

## Available settings

- [accept-flake-config](#)

Whether to accept nix configuration from a flake without prompting.

**Warning** This setting is part of an [experimental feature](#).

---

To change this setting, you need to make sure the corresponding experimental feature, `flakes`, is enabled. For example, include the following in `nix.conf`:

```
extra-experimental-features = flakes
accept-flake-config = ...
```

**Default:** `false`

- [access-tokens](#)

Access tokens used to access protected GitHub, GitLab, or other locations requiring token-based authentication.

Access tokens are specified as a string made up of space-separated `host=token` values. The specific token used is selected by matching the `host` portion against the "host" specification of the input. The actual use of the `token` value is determined by the type of resource being accessed:

- Github: the token value is the OAUTH-TOKEN string obtained as the Personal Access Token from the Github server (see <https://docs.github.com/en/developers/apps/building-oauth-apps/authorizing-oauth-apps>).
- Gitlab: the token value is either the OAuth2 token or the Personal Access Token (these are different types tokens for gitlab, see <https://docs.gitlab.com/12.10/ee/api/README.html#authentication>). The `token` value should be `type:tokenstring` where `type` is either `OAuth2` or `PAT` to indicate which type of token is being specified.

Example `~/.config/nix/nix.conf`:

```
access-tokens = github.com=23ac...b289
gitlab.mycompany.com=PAT:A123Bp_Cd..EfG gitlab.com=OAuth2:1jklw3jk
```

Example `~/code/flake.nix`:

```
input.foo = {
 type = "gitlab";
 host = "gitlab.mycompany.com";
 owner = "mycompany";
 repo = "pro";
};
```

This example specifies three tokens, one each for accessing github.com, gitlab.mycompany.com, and gitlab.com.

The `input.foo` uses the "gitlab" fetcher, which might require specifying the token type along with the token value.

**Default:** `empty`

- `allow-dirty`

Whether to allow dirty Git/Mercurial trees.

**Default:** `true`

- `allow-import-from-derivation`

By default, Nix allows you to `import` from a derivation, allowing building at evaluation time. With this option set to false, Nix will throw an error when evaluating an expression that uses this feature, allowing users to ensure their evaluation will not require any builds to take place.

**Default:** `true`

- `allow-new-privileges`

(Linux-specific.) By default, builders on Linux cannot acquire new privileges by calling `setuid/setgid` programs or programs that have file capabilities. For example, programs such as `sudo` or `ping` will fail. (Note that in sandbox builds, no such programs are available unless you bind-mount them into the sandbox via the `sandbox-paths` option.) You can allow the use of such programs by enabling this option. This is impure and usually undesirable, but may be useful in certain scenarios (e.g. to spin up containers or set up userspace network interfaces in tests).

**Default:** `false`

- `allow-symlinked-store`

If set to `true`, Nix will stop complaining if the store directory (typically `/nix/store`) contains symlink components.

This risks making some builds "impure" because builders sometimes "canonicalise" paths by resolving all symlink components. Problems occur if those builds are then deployed to machines where `/nix/store` resolves to a different location from that of the build machine. You can enable this setting if you are sure you're not going to do that.

**Default:** `false`

- `allow-unsafe-native-code-during-evaluation`

Whether builtin functions that allow executing native code should be enabled.

**Default:** `false`

- `allowed-impure-host-deps`

Which prefixes to allow derivations to ask for access to (primarily for Darwin).

**Default:** `empty`

- `allowed-uris`

A list of URI prefixes to which access is allowed in restricted evaluation mode. For example, when set to `https://github.com/NixOS`, builtin functions such as `fetchGit` are allowed to access `https://github.com/NixOS/patchelf.git`.

**Default:** `empty`

- `allowed-users`

A list user names, separated by whitespace. These users are allowed to connect to the Nix daemon.

You can specify groups by prefixing names with `@`. For instance, `@wheel` means all users in the `wheel` group. Also, you can allow all users by specifying `*`.

---

### Note

Trusted users (set in `trusted-users`) can always connect to the Nix daemon.

---

**Default:** `*`

- **auto-allocate-uids**

Whether to select UIDs for builds automatically, instead of using the users in `build-users-group`.

UIDs are allocated starting at 872415232 (0x34000000) on Linux and 56930 on macOS.

**Default:** `false`

- **auto-optimise-store**

If set to `true`, Nix automatically detects files in the store that have identical contents, and replaces them with hard links to a single copy. This saves disk space. If set to `false` (the default), you can still run `nix-store --optimise` to get rid of duplicate files.

**Default:** `false`

- **bash-prompt**

The bash prompt (`PS1`) in `nix develop` shells.

**Default:** `empty`

- **bash-prompt-prefix**

Prefix prepended to the `PS1` environment variable in `nix develop` shells.

**Default:** `empty`

- **bash-prompt-suffix**

Suffix appended to the `PS1` environment variable in `nix develop` shells.

**Default:** `empty`

- **build-hook**

The path to the helper program that executes remote builds.

Nix communicates with the build hook over `stdio` using a custom protocol to request builds that cannot be performed directly by the Nix daemon. The default value is the internal Nix binary that implements remote building.

---

**Important**

Change this setting only if you really know what you're doing.

---

**Default:** *empty*

- **build-poll-interval**

How often (in seconds) to poll for locks.

**Default:** 5

- **build-users-group**

This option specifies the Unix group containing the Nix build user accounts. In multi-user Nix installations, builds should not be performed by the Nix account since that would allow users to arbitrarily modify the Nix store and database by supplying specially crafted builders; and they cannot be performed by the calling user since that would allow him/her to influence the build result.

Therefore, if this option is non-empty and specifies a valid group, builds will be performed under the user accounts that are a member of the group specified here (as listed in `/etc/group`). Those user accounts should not be used for any other purpose!

Nix will never run two builds under the same user account at the same time. This is to prevent an obvious security hole: a malicious user writing a Nix expression that modifies the build result of a legitimate Nix expression being built by another user. Therefore it is good to have as many Nix build user accounts as you can spare. (Remember: uids are cheap.)

The build users should have permission to create files in the Nix store, but not delete them. Therefore, `/nix/store` should be owned by the Nix account, its group should be the group specified here, and its mode should be 1775 .

If the build users group is empty, builds will be performed under the uid of the Nix process (that is, the uid of the caller if `NIX_REMOTE` is empty, the uid under which the Nix daemon runs if `NIX_REMOTE` is `daemon`). Obviously, this should not be used with a nix daemon accessible to untrusted clients.

Defaults to `nixbld` when running as root, *empty* otherwise.

**Default:** *machine-specific*

- **builders**

A semicolon-separated list of build machines. For the exact format and examples, see

## the manual chapter on remote builds

**Default:** `@/dummy/machines`

- `builders-use-substitutes`

If set to `true`, Nix will instruct remote build machines to use their own binary substitutes if available. In practical terms, this means that remote hosts will fetch as many build dependencies as possible from their own substitutes (e.g, from `cache.nixos.org`), instead of waiting for this host to upload them all. This can drastically reduce build times if the network connection between this computer and the remote build host is slow.

**Default:** `false`

- `commit-lockfile-summary`

The commit summary to use when committing changed flake lock files. If empty, the summary is generated based on the action performed.

---

**Warning** This setting is part of an [experimental feature](#).

---

To change this setting, you need to make sure the corresponding experimental feature, `flakes`, is enabled. For example, include the following in `nix.conf`:

```
extra-experimental-features = flakes
commit-lockfile-summary = ...
```

**Default:** `empty`

- `compress-build-log`

If set to `true` (the default), build logs written to `/nix/var/log/nix/drvs` will be compressed on the fly using bzip2. Otherwise, they will not be compressed.

**Default:** `true`

**Deprecated alias:** `build-compress-log`

- `connect-timeout`

The timeout (in seconds) for establishing connections in the binary cache substituter. It corresponds to `curl`'s `--connect-timeout` option. A value of 0 means no limit.

**Default:** `0`

- `cores`

Sets the value of the `NIX_BUILD_CORES` environment variable in the invocation of builders. Builders can use this variable at their discretion to control the maximum amount of parallelism. For instance, in Nixpkgs, if the derivation attribute `enableParallelBuilding` is set to `true`, the builder passes the `-jN` flag to GNU Make. It can be overridden using the `--cores` command line switch and defaults to `1`. The value `0` means that the builder should use all available CPU cores in the system.

**Default:** *machine-specific***Deprecated alias:** `build-cores`

- `diff-hook`

Absolute path to an executable capable of diffing build results. The hook is executed if `run-diff-hook` is true, and the output of a build is known to not be the same. This program is not executed to determine if two results are the same.

The diff hook is executed by the same user and group who ran the build. However, the diff hook does not have write access to the store path just built.

The diff hook program receives three parameters:

1. A path to the previous build's results
2. A path to the current build's results
3. The path to the build's derivation
4. The path to the build's scratch directory. This directory will exist only if the build was run with `--keep-failed`.

The stderr and stdout output from the diff hook will not be displayed to the user. Instead, it will print to the nix-daemon's log.

When using the Nix daemon, `diff-hook` must be set in the `nix.conf` configuration file, and cannot be passed at the command line.

**Default:** ````

- `download-attempts`

How often Nix will attempt to download a file before giving up.

**Default:** 5

- `download-speed`

Specify the maximum transfer rate in kilobytes per second you want Nix to use for downloads.

**Default:** 0

- `eval-cache`

Whether to use the flake evaluation cache.

**Default:** true

- `experimental-features`

Experimental features that are enabled.

Example:

```
experimental-features = nix-command flakes
```

The following experimental features are available:

- `auto-allocate-uids`
- `ca-derivations`
- `cgroups`
- `daemon-trust-override`
- `dynamic-derivations`
- `fetch-closure`
- `flakes`
- `impure-derivations`
- `nix-command`
- `no-url-literals`
- `parse-toml-timestamps`
- `read-only-local-store`
- `recursive-nix`
- `repl-flake`

Experimental features are [further documented in the manual](#).

**Default:** empty

- **extra-platforms**

System types of executables that can be run on this machine.

Nix will only build a given [derivation](#) locally when its [system](#) attribute equals any of the values specified here or in the [system option](#).

Setting this can be useful to build derivations locally on compatible machines:

- [i686-linux](#) executables can be run on [x86\\_64-linux](#) machines (set by default)
- [x86\\_64-darwin](#) executables can be run on macOS [aarch64-darwin](#) with Rosetta 2 (set by default where applicable)
- [armv6](#) and [armv5tel](#) executables can be run on [armv7](#)
- some [aarch64](#) machines can also natively run 32-bit ARM code
- [qemu-user](#) may be used to support non-native platforms (though this may be slow and buggy)

Build systems will usually detect the target platform to be the current physical system and therefore produce machine code incompatible with what may be intended in the derivation. You should design your derivation's [builder](#) accordingly and cross-check the results when using this option against natively-built versions of your derivation.

**Default:** *machine-specific*

- **fallback**

If set to [true](#), Nix will fall back to building from source if a binary substitute fails. This is equivalent to the [--fallback](#) flag. The default is [false](#).

**Default:** [false](#)

**Deprecated alias:** [build-fallback](#)

- **filter-syccalls**

Whether to prevent certain dangerous system calls, such as creation of setuid/setgid files or adding ACLs or extended attributes. Only disable this if you're aware of the security implications.

**Default:** [true](#)

- **flake-registry**

Path or URI of the global flake registry.

When empty, disables the global flake registry.

---

**Warning** This setting is part of an [experimental feature](#).

---

To change this setting, you need to make sure the corresponding experimental feature, `flakes`, is enabled. For example, include the following in `nix.conf`:

```
extra-experimental-features = flakes
flake-registry = ...
```

**Default:** <https://channels.nixos.org/flake-registry.json>

- [fsync-metadata](#)

If set to `true`, changes to the Nix store metadata (in `/nix/var/nix/db`) are synchronously flushed to disk. This improves robustness in case of system crashes, but reduces performance. The default is `true`.

**Default:** `true`

- [gc-reserved-space](#)

Amount of reserved disk space for the garbage collector.

**Default:** `8388608`

- [hashed-mirrors](#)

A list of web servers used by `builtins.fetchurl` to obtain files by hash. Given a hash type `ht` and a base-16 hash `h`, Nix will try to download the file from `hashed-mirror/ht/h`. This allows files to be downloaded even if they have disappeared from their original URI. For example, given an example mirror <http://tarballs.nixos.org/>, when building the derivation

```
builtins.fetchurl {
 url = "https://example.org/foo-1.2.3.tar.xz";
 sha256 =
 "2c26b46b68fffc68ff99b453c1d30413413422d706483bfa0f98a5e886266e7ae";
}
```

Nix will attempt to download this file from <http://tarballs.nixos.org/sha256>

`/2c26b46b68ffc68ff99b453c1d30413413422d706483bfa0f98a5e886266e7ae` first. If it is not available there, it will try the original URI.

**Default:** `empty`

- `http-connections`

The maximum number of parallel TCP connections used to fetch files from binary caches and by other downloads. It defaults to 25. 0 means no limit.

**Default:** `25`

**Deprecated alias:** `binary-caches-parallel-connections`

- `http2`

Whether to enable HTTP/2 support.

**Default:** `true`

- `id-count`

The number of UIDs/GIDs to use for dynamic ID allocation.

**Default:** `8388608`

- `ignore-try`

If set to true, ignore exceptions inside 'tryEval' calls when evaluating nix expressions in debug mode (using the --debugger flag). By default the debugger will pause on all exceptions.

**Default:** `false`

- `ignored-acls`

A list of ACLs that should be ignored, normally Nix attempts to remove all ACLs from files and directories in the Nix store, but some ACLs like `security.selinux` or `system.nfs4_acl` can't be removed even by root. Therefore it's best to just ignore them.

**Default:** `security.csm security.selinux system.nfs4_acl`

- `imPERSONATE-linuX-26`

Whether to impersonate a Linux 2.6 machine on newer kernels.

**Default:** `false`

**Deprecated alias:** `build-impersonate-linux-26`

- `keep-build-log`

If set to `true` (the default), Nix will write the build log of a derivation (i.e. the standard output and error of its builder) to the directory `/nix/var/log/nix/drvs`. The build log can be retrieved using the command `nix-store -l path`.

**Default:** `true`

**Deprecated alias:** `build-keep-log`

- `keep-derivations`

If `true` (default), the garbage collector will keep the derivations from which non-garbage store paths were built. If `false`, they will be deleted unless explicitly registered as a root (or reachable from other roots).

Keeping derivation around is useful for querying and traceability (e.g., it allows you to ask with what dependencies or options a store path was built), so by default this option is on. Turn it off to save a bit of disk space (or a lot if `keep-outputs` is also turned on).

**Default:** `true`

**Deprecated alias:** `gc-keep-derivations`

- `keep-env-derivations`

If `false` (default), derivations are not stored in Nix user environments. That is, the derivations of any build-time-only dependencies may be garbage-collected.

If `true`, when you add a Nix derivation to a user environment, the path of the derivation is stored in the user environment. Thus, the derivation will not be garbage-collected until the user environment generation is deleted (`nix-env --delete-generations`). To prevent build-time-only dependencies from being collected, you should also turn on `keep-outputs`.

The difference between this option and `keep-derivations` is that this one is “sticky”: it applies to any user environment created while this option was enabled, while `keep-derivations` only applies at the moment the garbage collector is run.

**Default:** `false`

**Deprecated alias:** `env-keep-derivations`

- `keep-failed`

Whether to keep temporary directories of failed builds.

**Default:** `false`

- `keep-going`

Whether to keep building derivations when another build fails.

**Default:** `false`

- `keep-outputs`

If `true`, the garbage collector will keep the outputs of non-garbage derivations. If `false` (default), outputs will be deleted unless they are GC roots themselves (or reachable from other roots).

In general, outputs must be registered as roots separately. However, even if the output of a derivation is registered as a root, the collector will still delete store paths that are used only at build time (e.g., the C compiler, or source tarballs downloaded from the network). To prevent it from doing so, set this option to `true`.

**Default:** `false`**Deprecated alias:** `gc-keep-outputs`

- `log-lines`

The number of lines of the tail of the log to show if a build fails.

**Default:** `10`

- `max-build-log-size`

This option defines the maximum number of bytes that a builder can write to its `stdout/stderr`. If the builder exceeds this limit, it's killed. A value of `0` (the default) means that there is no limit.

**Default:** `0`**Deprecated alias:** `build-max-log-size`

- `max-free`

When a garbage collection is triggered by the `min-free` option, it stops as soon as `max-free` bytes are available. The default is infinity (i.e. delete all garbage).

**Default:** `-1`

- `max-jobs`

This option defines the maximum number of jobs that Nix will try to build in parallel. The default is `1`. The special value `auto` causes Nix to use the number of CPUs in your system. `0` is useful when using remote builders to prevent any local builds (except for `preferLocalBuild` derivation attribute which executes locally regardless). It can be overridden using the `--max-jobs` (`-j`) command line switch.

**Default:** `1`

**Deprecated alias:** `build-max-jobs`

- `max-silent-time`

This option defines the maximum number of seconds that a builder can go without producing any data on standard output or standard error. This is useful (for instance in an automated build system) to catch builds that are stuck in an infinite loop, or to catch remote builds that are hanging due to network problems. It can be overridden using the `--max-silent-time` command line switch.

The value `0` means that there is no timeout. This is also the default.

**Default:** `0`

**Deprecated alias:** `build-max-silent-time`

- `max-substitution-jobs`

This option defines the maximum number of substitution jobs that Nix will try to run in parallel. The default is `16`. The minimum value one can choose is `1` and lower values will be interpreted as `1`.

**Default:** `16`

**Deprecated alias:** `substitution-max-jobs`

- `min-free`

When free disk space in `/nix/store` drops below `min-free` during a build, Nix performs a garbage-collection until `max-free` bytes are available or there is no more

garbage. A value of `0` (the default) disables this feature.

**Default:** `0`

- `min-free-check-interval`

Number of seconds between checking free disk space.

**Default:** `5`

- `nar-buffer-size`

Maximum size of NARs before spilling them to disk.

**Default:** `33554432`

- `narinfo-cache-negative-ttl`

The TTL in seconds for negative lookups. If a store path is queried from a substituter but was not found, there will be a negative lookup cached in the local disk cache database for the specified duration.

**Default:** `3600`

- `narinfo-cache-positive-ttl`

The TTL in seconds for positive lookups. If a store path is queried from a substituter, the result of the query will be cached in the local disk cache database including some of the NAR metadata. The default TTL is a month, setting a shorter TTL for positive lookups can be useful for binary caches that have frequent garbage collection, in which case having a more frequent cache invalidation would prevent trying to pull the path again and failing with a hash mismatch if the build isn't reproducible.

**Default:** `2592000`

- `netrc-file`

If set to an absolute path to a `netrc` file, Nix will use the HTTP authentication credentials in this file when trying to download from a remote host through HTTP or HTTPS. Defaults to `$NIX_CONF_DIR/netrc`.

The `netrc` file consists of a list of accounts in the following format:

```
machine my-machine
login my-username
password my-password
```

For the exact syntax, see [the curl documentation](#).

---

## Note

This must be an absolute path, and `~` is not resolved. For example, `~/ .netrc` won't resolve to your home directory's `.netrc`.

---

**Default:** `/dummy/netrc`

- `nix-path`

List of directories to be searched for `<...>` file references

In particular, outside of [pure evaluation mode](#), this determines the value of `builtins.nixPath`.

**Default:** `empty`

- `plugin-files`

A list of plugin files to be loaded by Nix. Each of these files will be dlopened by Nix, allowing them to affect execution through static initialization. In particular, these plugins may construct static instances of `RegisterPrimOp` to add new primops or constants to the expression language, `RegisterStoreImplementation` to add new store implementations, `RegisterCommand` to add new subcommands to the `nix` command, and `RegisterSetting` to add new nix config settings. See the constructors for those types for more details.

Warning! These APIs are inherently unstable and may change from release to release.

Since these files are loaded into the same address space as Nix itself, they must be DSOs compatible with the instance of Nix running at the time (i.e. compiled against the same headers, not linked to any incompatible libraries). They should not be linked to any Nix libs directly, as those will be available already at load time.

If an entry in the list is a directory, all files in the directory are loaded as plugins (non-recursively).

**Default:** *empty*

- **post-build-hook**

Optional. The path to a program to execute after each build.

This option is only settable in the global `nix.conf`, or on the command line by trusted users.

When using the nix-daemon, the daemon executes the hook as `root`. If the nix-daemon is not involved, the hook runs as the user executing the nix-build.

- The hook executes after an evaluation-time build.
- The hook does not execute on substituted paths.
- The hook's output always goes to the user's terminal.
- If the hook fails, the build succeeds but no further builds execute.
- The hook executes synchronously, and blocks other builds from progressing while it runs.

The program executes with no arguments. The program's environment contains the following environment variables:

- `DRV_PATH` The derivation for the built paths.

Example: `/nix/store/5nihn1a7pa8b25l9zafqaqibznlvvp3f-bash-4.4-p23.drv`

- `OUT_PATHS` Output paths of the built derivation, separated by a space character.

Example: `/nix/store/zf5lbh336mnzf1nlswdn11g4n2m8zh3g-bash-4.4-p23-dev /nix/store/rjxwxwv1fpn9wa2x5ssk5phzwlcv4mna-bash-4.4-p23-doc /nix/store /6bqvbjkcp9695dq0dpl5y43nvy37pq1-bash-4.4-p23-info /nix/store /r7fng3kk3vlpdlh2idnrbn37vh4imlj2-bash-4.4-p23-man /nix/store /xfghy8ixrhz3kyy6p724iv3cxji088dx-bash-4.4-p23 .`

**Default:** *empty*

- **pre-build-hook**

If set, the path to a program that can set extra derivation-specific settings for this system. This is used for settings that can't be captured by the derivation model itself and are too variable between different versions of the same system to be hard-coded into nix.

The hook is passed the derivation path and, if sandboxes are enabled, the sandbox directory. It can then modify the sandbox and send a series of commands to modify various settings to stdout. The currently recognized commands are:

- `extra-sandbox-paths`

Pass a list of files and directories to be included in the sandbox for this build. One entry per line, terminated by an empty line. Entries have the same format as `sandbox-paths`.

**Default:** `empty`

- `preallocate-contents`

Whether to preallocate files when writing objects with known size.

**Default:** `false`

- `print-missing`

Whether to print what paths need to be built or downloaded.

**Default:** `true`

- `pure-eval`

Pure evaluation mode ensures that the result of Nix expressions is fully determined by explicitly declared inputs, and not influenced by external state:

- Restrict file system and network access to files specified by cryptographic hash
- Disable `builtins.currentSystem` and `builtins.currentTime`

**Default:** `false`

- `require-drop-supplementary-groups`

Following the principle of least privilege, Nix will attempt to drop supplementary groups when building with sandboxing.

However this can fail under some circumstances. For example, if the user lacks the `CAP_SETGID` capability. Search `setgroups(2)` for `EPERM` to find more detailed information on this.

If you encounter such a failure, setting this option to `false` will let you ignore it and continue. But before doing so, you should consider the security implications carefully. Not dropping supplementary groups means the build sandbox will be less restricted

than intended.

This option defaults to `true` when the user is root (since `root` usually has permissions to call `setgroups`) and `false` otherwise.

**Default:** `false`

- `require-sigs`

If set to `true` (the default), any non-content-addressed path added or copied to the Nix store (e.g. when substituting from a binary cache) must have a signature by a trusted key. A trusted key is one listed in `trusted-public-keys`, or a public key counterpart to a private key stored in a file listed in `secret-key-files`.

Set to `false` to disable signature checking and trust all non-content-addressed paths unconditionally.

(Content-addressed paths are inherently trustworthy and thus unaffected by this configuration option.)

**Default:** `true`

- `restrict-eval`

If set to `true`, the Nix evaluator will not allow access to any files outside of the Nix search path (as set via the `NIX_PATH` environment variable or the `-I` option), or to URLs outside of `allowed-uris`. The default is `false`.

**Default:** `false`

- `run-diff-hook`

If true, enable the execution of the `diff-hook` program.

When using the Nix daemon, `run-diff-hook` must be set in the `nix.conf` configuration file, and cannot be passed at the command line.

**Default:** `false`

- `sandbox`

If set to `true`, builds will be performed in a *sandboxed environment*, i.e., they're isolated from the normal file system hierarchy and will only see their dependencies in the Nix store, the temporary build directory, private versions of `/proc`, `/dev`, `/dev/shm` and `/dev/pts` (on Linux), and the paths configured with the `sandbox-paths` option. This is

useful to prevent undeclared dependencies on files in directories such as `/usr/bin`. In addition, on Linux, builds run in private PID, mount, network, IPC and UTS namespaces to isolate them from other processes in the system (except that fixed-output derivations do not run in private network namespace to ensure they can access the network).

Currently, sandboxing only work on Linux and macOS. The use of a sandbox requires that Nix is run as root (so you should use the “build users” feature to perform the actual builds under different users than root).

If this option is set to `relaxed`, then fixed-output derivations and derivations that have the `__noChroot` attribute set to `true` do not run in sandboxes.

The default is `true` on Linux and `false` on all other platforms.

**Default:** `true`

**Deprecated alias:** `build-use-chroot`, `build-use-sandbox`

- `sandbox-build-dir`

The build directory inside the sandbox.

**Default:** `/build`

- `sandbox-dev-shm-size`

This option determines the maximum size of the `tmpfs` filesystem mounted on `/dev/shm` in Linux sandboxes. For the format, see the description of the `size` option of `tmpfs` in `mount(8)`. The default is `50%`.

**Default:** `50%`

- `sandbox-fallback`

Whether to disable sandboxing when the kernel doesn't allow it.

**Default:** `true`

- `sandbox-paths`

A list of paths bind-mounted into Nix sandbox environments. You can use the syntax `target=source` to mount a path in a different location in the sandbox; for instance, `/bin=/nix-bin` will mount the path `/nix-bin` as `/bin` inside the sandbox. If `source` is followed by `?`, then it is not an error if `source` does not exist; for example,

`/dev/nvidiactl?` specifies that `/dev/nvidiactl` will only be mounted in the sandbox if it exists in the host filesystem.

If the source is in the Nix store, then its closure will be added to the sandbox as well.

Depending on how Nix was built, the default value for this option may be empty or provide `/bin/sh` as a bind-mount of `bash`.

**Default:** *empty*

**Deprecated alias:** `build-chroot-dirs`, `build-sandbox-paths`

- `secret-key-files`

A whitespace-separated list of files containing secret (private) keys. These are used to sign locally-built paths. They can be generated using `nix-store --generate-binary-cache-key`. The corresponding public key can be distributed to other users, who can add it to `trusted-public-keys` in their `nix.conf`.

**Default:** *empty*

- `show-trace`

Whether Nix should print out a stack trace in case of Nix expression evaluation errors.

**Default:** `false`

- `ssl-cert-file`

The path of a file containing CA certificates used to authenticate `https://` downloads. Nix by default will use the first of the following files that exists:

1. `/etc/ssl/certs/ca-certificates.crt`
2. `/nix/var/nix/profiles/default/etc/ssl/certs/ca-bundle.crt`

The path can be overridden by the following environment variables, in order of precedence:

1. `NIX_SSL_CERT_FILE`
2. `SSL_CERT_FILE`

**Default:** *empty*

- `stalled-download-timeout`

The timeout (in seconds) for receiving data from servers during download. Nix cancels

idle downloads after this timeout's duration.

**Default:** `300`

- `start-id`

The first UID and GID to use for dynamic ID allocation.

**Default:** `872415232`

- `store`

The [URL of the Nix store](#) to use for most operations. See `nix help-stores` for supported store types and settings.

**Default:** `auto`

- `substitute`

If set to `true` (default), Nix will use binary substitutes if available. This option can be disabled to force building from source.

**Default:** `true`

**Deprecated alias:** `build-use-substitutes`

- `substituters`

A list of [URLs of Nix stores](#) to be used as substituters, separated by whitespace. A substituter is an additional [store]{..../glossary.md##gloss-store} from which Nix can obtain `store objects` instead of building them.

Substituters are tried based on their priority value, which each substituter can set independently. Lower value means higher priority. The default is <https://cache.nixos.org>, which has a priority of 40.

At least one of the following conditions must be met for Nix to use a substituter:

- The substituter is in the `trusted-substituters` list
- The user calling Nix is in the `trusted-users` list

In addition, each store path should be trusted as described in `trusted-public-keys`

**Default:** <https://cache.nixos.org/>

**Deprecated alias:** `binary-caches`

- [sync-before-registering](#)

Whether to call `sync()` before registering a path as valid.

**Default:** `false`

- [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`

- [system-features](#)

A set of system “features” supported by this machine, e.g. `kvm`. Derivations can express a dependency on such features through the derivation attribute `requiredSystemFeatures`. For example, the attribute

```
requiredSystemFeatures = ["kvm"];
```

ensures that the derivation can only be built on a machine with the `kvm` feature.

This setting by default includes `kvm` if `/dev/kvm` is accessible, `apple-virt` if hardware virtualization is available on macOS, and the pseudo-features `nixos-test`, `benchmark`

and `big-parallel` that are used in Nixpkgs to route builds to specific machines.

**Default:** *machine-specific*

- `tarball-ttl`

The number of seconds a downloaded tarball is considered fresh. If the cached tarball is stale, Nix will check whether it is still up to date using the ETag header. Nix will download a new version if the ETag header is unsupported, or the cached ETag doesn't match.

Setting the TTL to `0` forces Nix to always check if the tarball is up to date.

Nix caches tarballs in `$XDG_CACHE_HOME/nix/tarballs`.

Files fetched via `NIX_PATH`, `fetchGit`, `fetchMercurial`, `fetchTarball`, and `fetchurl` respect this TTL.

**Default:** `3600`

- `timeout`

This option defines the maximum number of seconds that a builder can run. This is useful (for instance in an automated build system) to catch builds that are stuck in an infinite loop but keep writing to their standard output or standard error. It can be overridden using the `--timeout` command line switch.

The value `0` means that there is no timeout. This is also the default.

**Default:** `0`

**Deprecated alias:** `build-timeout`

- `trace-function-calls`

If set to `true`, the Nix evaluator will trace every function call. Nix will print a log message at the "vomit" level for every function entrance and function exit.

```
function-trace entered undefined position at 1565795816999559622
function-trace exited undefined position at 1565795816999581277
function-trace entered /nix/store/.../example.nix:226:41 at
1565795253249935150
function-trace exited /nix/store/.../example.nix:226:41 at
1565795253249941684
```

The `undefined position` means the function call is a builtin.

Use the `contrib/stack-collapse.py` script distributed with the Nix source code to convert the trace logs in to a format suitable for `flamegraph.pl`.

**Default:** `false`

- `trace-verbose`

Whether `builtins.traceVerbose` should trace its first argument when evaluated.

**Default:** `false`

- `trusted-public-keys`

A whitespace-separated list of public keys.

At least one of the following condition must be met for Nix to accept copying a store object from another Nix store (such as a substituter):

- the store object has been signed using a key in the trusted keys list
- the `require-sigs` option has been set to `false`
- the store object is `output-addressed`

**Default:** `cache.nixos.org-1:6NCHdD59X431o0gWypbMrAURkbJ16ZPMQFGspcDShjY=`

**Deprecated alias:** `binary-cache-public-keys`

- `trusted-substituters`

A list of [Nix store URLs](#), separated by whitespace. These are not used by default, but users of the Nix daemon can enable them by specifying `substituters`.

Unprivileged users (those set in only `allowed-users` but not `trusted-users`) can pass as `substituters` only those URLs listed in `trusted-substituters`.

**Default:** `empty`

**Deprecated alias:** `trusted-binary-caches`

- `trusted-users`

A list of user names, separated by whitespace. These users will have additional rights when connecting to the Nix daemon, such as the ability to specify additional `substituters`, or to import unsigned `NARs`.

You can also specify groups by prefixing names with `@`. For instance, `@wheel` means all users in the `wheel` group.

---

## Warning

Adding a user to `trusted-users` is essentially equivalent to giving that user root access to the system. For example, the user can access or replace store path contents that are critical for system security.

---

### **Default:** `root`

- `use-case-hack`

Whether to enable a Darwin-specific hack for dealing with file name collisions.

### **Default:** `false`

- `use-cgroups`

Whether to execute builds inside cgroups. This is only supported on Linux.

Cgroups are required and enabled automatically for derivations that require the `uid-range` system feature.

### **Default:** `false`

- `use-registries`

Whether to use flake registries to resolve flake references.

---

**Warning** This setting is part of an [experimental feature](#).

---

To change this setting, you need to make sure the corresponding experimental feature, `flakes`, is enabled. For example, include the following in `nix.conf`:

```
extra-experimental-features = flakes
use-registries = ...
```

### **Default:** `true`

- `use-sqlite-wal`

Whether SQLite should use WAL mode.

**Default:** `true`

- `use-xdg-base-directories`

If set to `true`, Nix will conform to the [XDG Base Directory Specification](#) for files in `$HOME`. The environment variables used to implement this are documented in the [Environment Variables section](#).

---

**Warning** This changes the location of some well-known symlinks that Nix creates, which might break tools that rely on the old, non-XDG-conformant locations.

---

In particular, the following locations change:

| Old                          | New                                        |
|------------------------------|--------------------------------------------|
| <code>~/.nix-profile</code>  | <code>\$XDG_STATE_HOME/nix/profile</code>  |
| <code>~/.nix-defexpr</code>  | <code>\$XDG_STATE_HOME/nix/defexpr</code>  |
| <code>~/.nix-channels</code> | <code>\$XDG_STATE_HOME/nix/channels</code> |

If you already have Nix installed and are using [profiles](#) or [channels](#), you should migrate manually when you enable this option. If `$XDG_STATE_HOME` is not set, use `$HOME/.local/state/nix` instead of `$XDG_STATE_HOME/nix`. This can be achieved with the following shell commands:

```
nix_state_home=${XDG_STATE_HOME-$HOME/.local/state}/nix
mkdir -p $nix_state_home
mv $HOME/.nix-profile $nix_state_home/profile
mv $HOME/.nix-defexpr $nix_state_home/defexpr
mv $HOME/.nix-channels $nix_state_home/channels
```

**Default:** `false`

- `user-agent-suffix`

String appended to the user agent in HTTP requests.

**Default:** `empty`

- `warn-dirty`

Whether to warn about dirty Git/Mercurial trees.

**Default:** `true`

# 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
/profile -> profile-7-link
lrwxrwxrwx 1 alice users 51 Oct 28 16:18 /home/alice/.local/state/nix/profiles
/profile-5-link -> /nix/store/q69xad13ghpf7ir87h0b2gd28lafjj1j-profile
lrwxrwxrwx 1 alice users 51 Oct 29 13:20 /home/alice/.local/state/nix/profiles
/profile-6-link -> /nix/store/6bvhpysd7vwz7k3b0pndn7ifi5xr32dg-profile
lrwxrwxrwx 1 alice users 51 Nov 25 14:35 /home/alice/.local/state/nix/profiles
/profile-7-link -> /nix/store/mp0x6xnsg0b8qhswy6riqvimai4gm677-profile
```

Each of these symlinks is a root for the Nix garbage collector.

The contents of the store path corresponding to each version of the profile is a tree of symlinks to the files of the installed packages, e.g.

```
$ ll -R ~eelco/.local/state/nix/profiles/profile-7-link/
/home/eelco/.local/state/nix/profiles/profile-7-link/:
total 20
dr-xr-xr-x 2 root root 4096 Jan 1 1970 bin
-r--r--r-- 2 root root 1402 Jan 1 1970 manifest.nix
dr-xr-xr-x 4 root root 4096 Jan 1 1970 share

/home/eelco/.local/state/nix/profiles/profile-7-link/bin:
total 20
lwxrwxrwx 5 root root 79 Jan 1 1970 chromium -> /nix/store
/ijm5k0zqisvkdwjkc77mb9qzb35xfi4m-chromium-86.0.4240.111/bin/chromium
lwxrwxrwx 7 root root 87 Jan 1 1970 spotify -> /nix/store
/w9182874m1bl56smpr3m5zjj36jhp3rn-spotify-1.1.26.501.gbe1le53b-15/bin/spotify
lwxrwxrwx 3 root root 79 Jan 1 1970 zoom-us -> /nix/store
/wbhg2ga8f3h87s9h5k0slxk0m81m4cxl-zoom-us-5.3.469451.0927/bin/zoom-us

/home/eelco/.local/state/nix/profiles/profile-7-link/share/applications:
total 12
lwxrwxrwx 4 root root 120 Jan 1 1970 chromium-browser.desktop -> /nix/store
/4cf803y4vzfm3gyk3vzhzb2327v0kl8a-chromium-unwrapped-86.0.4240.111/share
/applications/chromium-browser.desktop
lwxrwxrwx 7 root root 110 Jan 1 1970 spotify.desktop -> /nix/store
/w9182874m1bl56smpr3m5zjj36jhp3rn-spotify-1.1.26.501.gbe1le53b-15/share
/applications/spotify.desktop
lwxrwxrwx 3 root root 107 Jan 1 1970 us.zoom.Zoom.desktop -> /nix/store
/wbhg2ga8f3h87s9h5k0slxk0m81m4cxl-zoom-us-5.3.469451.0927/share/applications
/us.zoom.Zoom.desktop

...
```

Each profile version contains a manifest file:

- `manifest.nix` used by `nix-env`.
- `manifest.json` used by `nix profile` (experimental).

## User profile link

A symbolic link to the user's current profile:

- `~/.nix-profile`
- `$XDG_STATE_HOME/nix/profile` if `use-xdg-base-directories` is set to `true`.

By default, this symlink points to:

- `$XDG_STATE_HOME/nix/profiles/profile` for regular users
- `$NIX_STATE_DIR/profiles/per-user/root/profile` for `root`

The `PATH` environment variable should include `/bin` subdirectory of the profile link (e.g.

`~/.nix-profile/bin`) for the user environment to be visible to the user. The [installer](#) sets this up by default, unless you enable [`use-xdg-base-directories`](#).

## manifest.nix

The manifest file records the provenance of the packages that are installed in a [profile](#) managed by [nix-env](#).

Here is an example of how this file might look like after installing [hello](#) from Nixpkgs:

```
[{
 meta = {
 available = true;
 broken = false;
 changelog =
 "https://git.savannah.gnu.org/cgit/Hello.git/plain/NEWS?h=v2.12.1";
 description = "A program that produces a familiar, friendly greeting";
 homepage = "https://www.gnu.org/software/Hello/manual/";
 insecure = false;
 license = {
 deprecated = false;
 free = true;
 fullName = "GNU General Public License v3.0 or later";
 redistributable = true;
 shortName = "gpl3Plus";
 SPDXId = "GPL-3.0-or-later";
 url = "https://spdx.org/licenses/GPL-3.0-or-later.html";
 };
 longDescription = ''
 GNU Hello is a program that prints "Hello, world!" when you run it.
 It is fully customizable.
 '';
 maintainers = [{{
 email = "edolstra+nixpkgs@gmail.com";
 github = "edolstra";
 githubId = 1148549;
 name = "Eelco Dolstra";
 }}];
 name = "hello-2.12.1";
 outputsToInstall = ["out"];
 platforms = [
 "i686-cygwin"
 "x86_64-cygwin"
 "x86_64-darwin"
 "i686-darwin"
 "aarch64-darwin"
 "armv7a-darwin"
 "i686-freebsd13"
 "x86_64-freebsd13"
 "aarch64-genode"
 "i686-genode"
 "x86_64-genode"
 "x86_64-solaris"
 "js-ghcjs"
 "aarch64-linux"
 "armv5tel-linux"
 "armv6l-linux"
 "armv7a-linux"
 "armv7l-linux"
 "i686-linux"
 "m68k-linux"
 "microblaze-linux"
]
 }
}]
```

```
"microblazeel-linux"
"mipsel-linux"
"mips64el-linux"
"powerpc64-linux"
"powerpc64le-linux"
"riscv32-linux"
"riscv64-linux"
"s390-linux"
"s390x-linux"
"x86_64-linux"
"mmix-mmixware"
"aarch64-netbsd"
"armv6l-netbsd"
"armv7a-netbsd"
"armv7l-netbsd"
"i686-netbsd"
"m68k-netbsd"
"mipsel-netbsd"
"powerpc-netbsd"
"riscv32-netbsd"
"riscv64-netbsd"
"x86_64-netbsd"
"aarch64_be-none"
"aarch64-none"
"arm-none"
"armv6l-none"
"avr-none"
"i686-none"
"microblaze-none"
"microblazeel-none"
"msp430-none"
"or1k-none"
"m68k-none"
"powerpc-none"
"powerpcle-none"
"riscv32-none"
"riscv64-none"
"rx-none"
"s390-none"
"s390x-none"
"vc4-none"
"x86_64-none"
"i686-openbsd"
"x86_64-openbsd"
"x86_64-redox"
"wasm64-wasi"
"wasm32-wasi"
"x86_64-windows"
"i686-windows"
];
position =
"/nix/store/7niq32w715567hbph0q13m5lqna64c1s-nixos-unstable.tar.gz/nixos-
unstable.tar.gz/pkgs/applications/misc/hello/default.nix:34";
```

```
unfree = false;
unsupported = false;
};
name = "hello-2.12.1";
out = {
 outPath = "/nix/store/260q5867crm1xjs4khgqpl6vr9kywql1-hello-2.12.1";
};
outPath = "/nix/store/260q5867crm1xjs4khgqpl6vr9kywql1-hello-2.12.1";
outputs = ["out"];
system = "x86_64-linux";
type = "derivation";
}]
```

Each element in this list corresponds to an installed package. It incorporates some attributes of the original derivation, including `meta`, `name`, `out`, `outPath`, `outputs`, `system`. This information is used by Nix for querying and updating the package.

## manifest.json

The manifest file records the provenance of the packages that are installed in a [profile](#) managed by [nix profile](#) (experimental).

Here is an example of what the file might look like after installing `zoom-us` from Nixpkgs:

```
{
 "version": 1,
 "elements": [
 {
 "active": true,
 "attrPath": "legacyPackages.x86_64-linux.zoom-us",
 "originalUrl": "flake:nixpkgs",
 "storePaths": [
 "/nix/store/wbhg2ga8f3h87s9h5k0slxk0m81m4cxl-zoom-us-5.3.469451.0927"
],
 "uri": "github:NixOS/nixpkgs/13d0c311e3ae923a00f734b43fd1d35b47d8943a"
 },
 ...
]
}
```

Each object in the array `elements` denotes an installed package and has the following fields:

- `originalUrl`: The [flake reference](#) specified by the user at the time of installation (e.g. `nixpkgs`). This is also the flake reference that will be used by [nix profile upgrade](#).
- `uri`: The locked flake reference to which `originalUrl` resolved.
- `attrPath`: The flake output attribute that provided this package. Note that this is not necessarily the attribute that the user specified, but the one resulting from applying the default attribute paths and prefixes; for instance, `hello` might resolve to `packages.x86_64-linux.hello` and the empty string to `packages.x86_64-linux.default`.
- `storePath`: The paths in the Nix store containing the package.
- `active`: Whether the profile contains symlinks to the files of this package. If set to false, the package is kept in the Nix store, but is not "visible" in the profile's symlink tree.

# Channels

A directory containing symlinks to Nix channels, managed by `nix-channel`:

- `$XDG_STATE_HOME/nix/profiles/channels` for regular users
- `$NIX_STATE_DIR/profiles/per-user/root/channels` for `root`

`nix-channel` uses a `profile` to store channels. This profile contains symlinks to the contents of those channels.

## Subscribed channels

The list of subscribed channels is stored in

- `~/.nix-channels`
- `$XDG_STATE_HOME/nix/channels` if `use-xdg-base-directories` is set to `true`

in the following format:

```
<url> <name>
...
...
```

# Default Nix expression

The source for the default [Nix expressions](#) used by `nix-env`:

- `~/.nix-defexpr`
- `$XDG_STATE_HOME/nix/defexpr` if `use-xdg-base-directories` is set to `true`.

It is loaded as follows:

- If the default expression is a file, it is loaded as a Nix expression.
- If the default expression is a directory containing a `default.nix` file, that `default.nix` file is loaded as a Nix expression.
- If the default expression is a directory without a `default.nix` file, then its contents (both files and subdirectories) are loaded as Nix expressions. The expressions are combined into a single attribute set, each expression under an attribute with the same name as the original file or subdirectory. Subdirectories without a `default.nix` file are traversed recursively in search of more Nix expressions, but the names of these intermediate directories are not added to the attribute paths of the default Nix expression.

Then, the resulting expression is interpreted like this:

- If the expression is an attribute set, it is used as the default Nix expression.
- If the expression is a function, an empty set is passed as argument and the return value is used as the default Nix expression.

For example, if the default expression contains two files, `foo.nix` and `bar.nix`, then the default Nix expression will be equivalent to

```
{
 foo = import ~/.nix-defexpr/foo.nix;
 bar = import ~/.nix-defexpr/bar.nix;
}
```

The file `manifest.nix` is always ignored.

The command `nix-channel` places a symlink to the user's current `channels profile` in this directory. This makes all subscribed channels available as attributes in the default expression.

## User channel link

A symlink that ensures that `nix-env` can find your channels:

- `~/.nix-defexpr/channels`
- `$XDG_STATE_HOME/defexpr/channels` if `use-xdg-base-directories` is set to `true`.

This symlink points to:

- `$XDG_STATE_HOME/profiles/channels` for regular users
- `$NIX_STATE_DIR/profiles/per-user/root/channels` for `root`

In a multi-user installation, you may also have `~/.nix-defexpr/channels_root`, which links to the channels of the root user. [nix-env](#) : .../nix-env.md

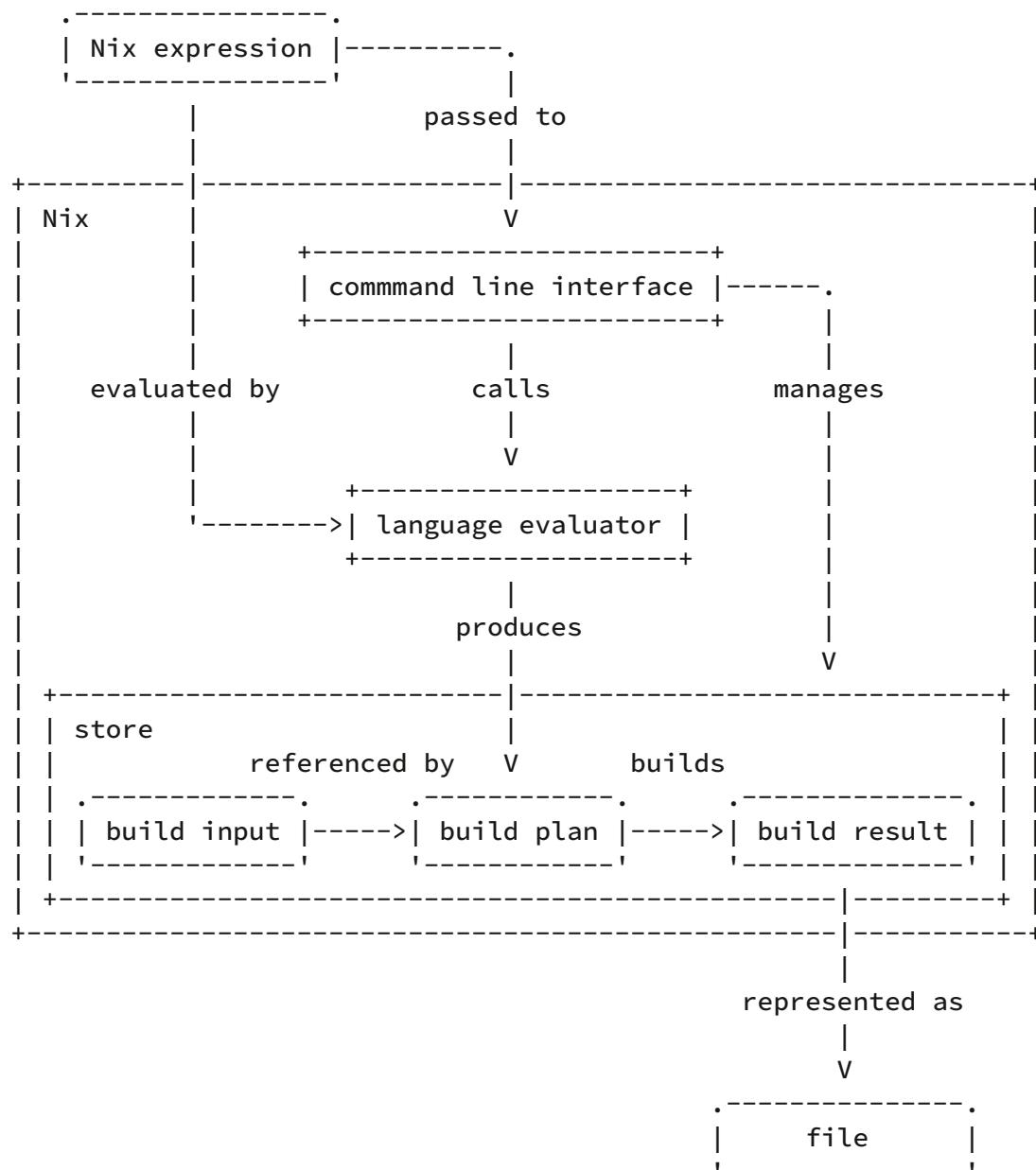
# Architecture

This chapter describes how Nix works. It should help users understand why Nix behaves as it does, and it should help developers understand how to modify Nix and how to write similar tools.

## Overview

Nix consists of [hierarchical layers](#).

The following [concept map](#) shows its main components (rectangles), the objects they operate on (rounded rectangles), and their interactions (connecting phrases):



At the top is the [command line interface](#) that drives the underlying layers.

The [Nix language](#) evaluator transforms Nix expressions into self-contained *build plans*, which are used to derive *build results* from referenced *build inputs*.

The command line interface and Nix expressions are what users deal with most.

**Note** The Nix language itself does not have a notion of *packages* or *configurations*. As far as we are concerned here, the inputs and results of a build plan are just data.

Underlying the command line interface and the Nix language evaluator is the [Nix store](#), a

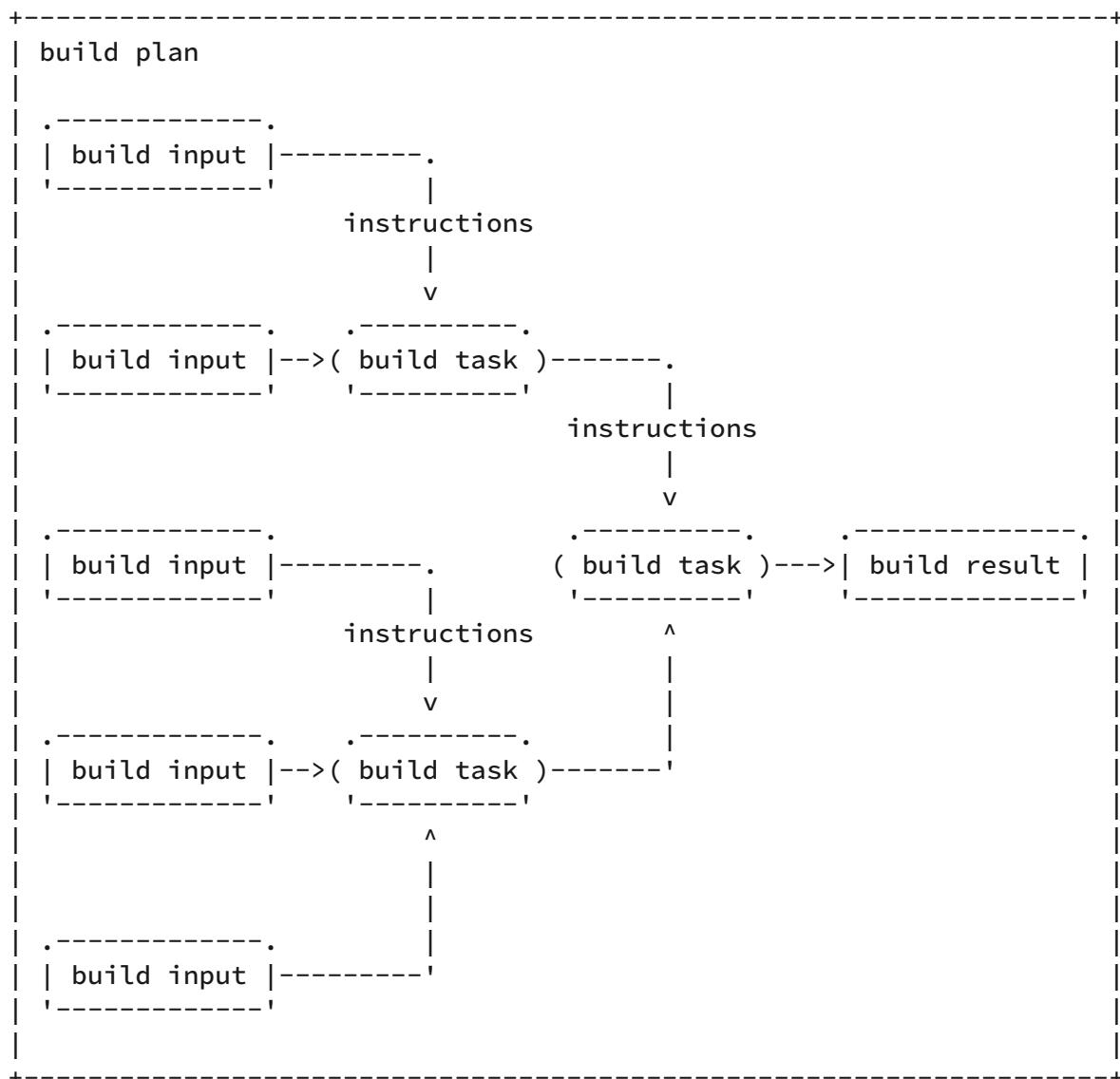
mechanism to keep track of build plans, data, and references between them. It can also execute build plans to produce new data, which are made available to the operating system as files.

A build plan itself is a series of *build tasks*, together with their build inputs.

**Important** A build task in Nix is called [derivation](#).

Each build task has a special build input executed as *build instructions* in order to perform the build. The result of a build task can be input to another build task.

The following [data flow diagram](#) shows a build plan for illustration. Build inputs used as instructions to a build task are marked accordingly:



# File System Object

Nix uses a simplified model of the file system, which consists of file system objects. Every file system object is one of the following:

- File
  - A possibly empty sequence of bytes for contents
  - A single boolean representing the [executable](#) permission

- Directory

Mapping of names to child file system objects

- [Symbolic link](#)

An arbitrary string. Nix does not assign any semantics to symbolic links.

File system objects and their children form a tree. A bare file or symlink can be a root file system object.

Nix does not encode any other file system notions such as [hard links](#), [permissions](#), timestamps, or other metadata.

## Examples of file system objects

A plain file:

```
50 B, executable: false
```

An executable file:

```
122 KB, executable: true
```

A symlink:

```
-> /usr/bin/sh
```

A directory with contents:

```
└── bin
 └── hello: 35 KB, executable: true
└── share
 ├── info
 │ └── hello.info: 36 KB, executable: false
 └── man
 └── man1
 └── hello.1.gz: 790 B, executable: false
```

A directory that contains a symlink and other directories:

```
└── bin -> share/go/bin
└── nix-support/
└── share/
```

# Protocols

This chapter documents various developer-facing interfaces provided by Nix.

# Lockable HTTP Tarball Protocol

Tarball flakes can be served as regular tarballs via HTTP or the file system (for `file://` URLs). Unless the server implements the Lockable HTTP Tarball protocol, it is the responsibility of the user to make sure that the URL always produces the same tarball contents.

An HTTP server can return an "immutable" HTTP URL appropriate for lock files. This allows users to specify a tarball flake input in `flake.nix` that requests the latest version of a flake (e.g. `https://example.org/hello/latest.tar.gz`), while `flake.lock` will record a URL whose contents will not change (e.g. `https://example.org/hello/<revision>.tar.gz`). To do so, the server must return an `HTTP Link header` with the `rel` attribute set to `immutable`, as follows:

```
Link: <flakeref>; rel="immutable"
```

(Note the required `<` and `>` characters around `flakeref`.)

`flakeref` must be a tarball flakeref. It can contain the tarball flake attributes `narHash`, `rev`, `revCount` and `lastModified`. If `narHash` is included, its value must be the NAR hash of the unpacked tarball (as computed via `nix hash path`). Nix checks the contents of the returned tarball against the `narHash` attribute. The `rev` and `revCount` attributes are useful when the tarball flake is a mirror of a fetcher type that has those attributes, such as Git or GitHub. They are not checked by Nix.

```
Link: <https://example.org/hello
/442793d9ec0584f6a6e82fa253850c8085bb150a.tar.gz
?rev=442793d9ec0584f6a6e82fa253850c8085bb150a
&revCount=835
&narHash=sha256-GUm8Uh/U74zFCwkvt9Mri4DSM%2BmHj3tYhXUkYpiv31M%3D>;
rel="immutable"
```

(The linebreaks in this example are for clarity and must not be included in the actual response.)

For tarball flakes, the value of the `lastModified` flake attribute is defined as the timestamp of the newest file inside the tarball.

# Derivation "ATerm" file format

For historical reasons, [derivations](#) are stored on-disk in [ATerm](#) format.

Derivations are serialised in one of the following formats:

- `Derive(...)`  
For all stable derivations.
- `DrvWithVersion(<version-string>, ...)`

The only `version-string`s that are in use today are for [experimental features](#):

- `"xp-dyn-drv"` for the [dynamic-derivations](#) experimental feature.

# Glossary

- [derivation](#)

A description of a build task. The result of a derivation is a store object. Derivations are typically specified in Nix expressions using the [derivation primitive](#). These are translated into low-level *store derivations* (implicitly by [nix-env](#) and [nix-build](#), or explicitly by [nix-instantiate](#)).

- [store derivation](#)

A [derivation](#) represented as a [.drv](#) file in the [store](#). It has a [store path](#), like any [store object](#).

Example: `/nix/store/g946hc4c8mdvq2g8vxx42z51qb71rvp-git-2.38.1.drv`

See [nix derivation show](#) (experimental) for displaying the contents of store derivations.

- [instantiate](#), instantiation

Translate a [derivation](#) into a [store derivation](#).

See [nix-instantiate](#).

- [realise](#), realisation

Ensure a [store path](#) is [valid](#).

This means either running the [builder](#) executable as specified in the corresponding [derivation](#), or fetching a pre-built [store object](#) from a [substituter](#), or delegating to a [remote builder](#) and retrieving the outputs.

See [nix-build](#) and [nix-store --realise](#).

See [nix build](#) (experimental).

- [content-addressed derivation](#)

A derivation which has the [\\_\\_contentAddressed](#) attribute set to [true](#).

- [fixed-output derivation](#)

A derivation which includes the [outputHash](#) attribute.

- [store](#)

The location in the file system where store objects live. Typically `/nix/store`.

From the perspective of the location where Nix is invoked, the Nix store can be referred to as a "*local*" or a "*remote*" one:

- A [local store](#) exists on the filesystem of the machine where Nix is invoked. You can use other local stores by passing the `--store` flag to the `nix` command. Local stores can be used for building derivations.
- A *remote store* exists anywhere other than the local filesystem. One example is the `/nix/store` directory on another machine, accessed via `ssh` or served by the `nix-serve` Perl script.

- [chroot store](#)

A [local store](#) whose canonical path is anything other than `/nix/store`.

- [binary cache](#)

A *binary cache* is a Nix store which uses a different format: its metadata and signatures are kept in `.narinfo` files rather than in a [Nix database](#). This different format simplifies serving store objects over the network, but cannot host builds. Examples of binary caches include S3 buckets and the [NixOS binary cache](#).

- [store path](#)

The location of a [store object](#) in the file system, i.e., an immediate child of the Nix store directory.

Example: `/nix/store/a040m110amc4h71lds2jmr8qrkj2jhxd-git-2.38.1`

- [file system object](#)

The Nix data model for representing simplified file system data.

See [File System Object](#) for details.

- [store object](#)

A store object consists of a [file system object](#), [references](#) to other store objects, and other metadata. It can be referred to by a [store path](#).

- [input-addressed store object](#)

A store object produced by building a non-[content-addressed](#), non-[fixed-output](#) derivation.

- [output-addressed store object](#)

A [store object](#) whose [store path](#) is determined by its contents. This includes derivations, the outputs of [content-addressed derivations](#), and the outputs of [fixed-output derivations](#).

- [substitute](#)

A substitute is a command invocation stored in the [Nix database](#) that describes how to build a store object, bypassing the normal build mechanism (i.e., derivations). Typically, the substitute builds the store object by downloading a pre-built version of the store object from some server.

- [substituter](#)

An additional [store](#) from which Nix can obtain store objects instead of building them. Often the substituter is a [binary cache](#), but any store can serve as substituter.

See the [substituters configuration option](#) for details.

- [purity](#)

The assumption that equal Nix derivations when run always produce the same output. This cannot be guaranteed in general (e.g., a builder can rely on external inputs such as the network or the system time) but the Nix model assumes it.

- [Nix database](#)

An SQLite database to track [references](#) between [store object](#)s. This is an implementation detail of the [local store](#).

Default location: `/nix/var/nix/db`.

- [Nix expression](#)

A high-level description of software packages and compositions thereof. Deploying software using Nix entails writing Nix expressions for your packages. Nix expressions are translated to derivations that are stored in the Nix store. These derivations can then be built.

- [reference](#)

A [store object](#) `o` is said to have a *reference* to a store object `P` if a [store path](#) to `P`

appears in the contents of `o`.

Store objects can refer to both other store objects and themselves. References from a store object to itself are called *self-references*. References other than a self-reference must not form a cycle.

- [reachable](#)

A store path `Q` is reachable from another store path `P` if `Q` is in the *closure* of the *references* relation.

- [closure](#)

The closure of a store path is the set of store paths that are directly or indirectly “reachable” from that store path; that is, it’s the closure of the path under the *references* relation. For a package, the closure of its derivation is equivalent to the build-time dependencies, while the closure of its output path is equivalent to its runtime dependencies. For correct deployment it is necessary to deploy whole closures, since otherwise at runtime files could be missing. The command `nix-store --query --requisites` prints out closures of store paths.

As an example, if the [store object](#) at path `P` contains a [reference](#) to a store object at path `Q`, then `Q` is in the closure of `P`. Further, if `Q` references `R` then `R` is also in the closure of `P`.

- [output](#)

A [store object](#) produced by a [derivation](#).

- [output path](#)

The [store path](#) to the [output](#) of a [derivation](#).

- [deriver](#)

The [store derivation](#) that produced an [output path](#).

- [validity](#)

A store path is valid if all [store objects](#) in its [closure](#) can be read from the [store](#).

For a [local store](#), this means:

- The store path leads to an existing [store object](#) in that [store](#).
- The store path is listed in the [Nix database](#) as being valid.
- All paths in the store path’s [closure](#) are valid.

- [user environment](#)

An automatically generated store object that consists of a set of symlinks to “active” applications, i.e., other store paths. These are generated automatically by [nix-env](#). See [profiles](#).

- [profile](#)

A symlink to the current *user environment* of a user, e.g., [/nix/var/nix/profiles/default](#).

- [installable](#)

Something that can be realised in the Nix store.

See [installables](#) for [nix commands](#) (experimental) for details.

- [NAR](#)

A *Nix ARchive*. This is a serialisation of a path in the Nix store. It can contain regular files, directories and symbolic links. NARs are generated and unpacked using [nix-store --dump](#) and [nix-store --restore](#).

- $\emptyset$

The empty set symbol. In the context of profile history, this denotes a package is not present in a particular version of the profile.

- $\varepsilon$

The epsilon symbol. In the context of a package, this means the version is empty. More precisely, the derivation does not have a version attribute.

- [string interpolation](#)

Expanding expressions enclosed in  `${ }`  within a [string](#), [path](#), or [attribute name](#).

See [String interpolation](#) for details.

- [experimental feature](#)

Not yet stabilized functionality guarded by named experimental feature flags. These flags are enabled or disabled with the [experimental-features](#) setting.

See the contribution guide on the [purpose and lifecycle of experimental feaures](#).

# Contributing

# Hacking

This section provides some notes on how to hack on Nix. To get the latest version of Nix from GitHub:

```
$ git clone https://github.com/NixOS/nix.git
$ cd nix
```

The following instructions assume you already have some version of Nix installed locally, so that you can use it to set up the development environment. If you don't have it installed, follow the [installation instructions](#).

## Building Nix with flakes

This section assumes you are using Nix with the `flakes` and `nix-command` experimental features enabled. See the [Building Nix](#) section for equivalent instructions using stable Nix interfaces.

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

```
$ nix develop
```

This shell also adds `./outputs/bin/nix` to your `$PATH` so you can run `nix` immediately after building it.

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

```
$ nix develop .#native-clang11StdenvPackages
```

---

### Note

Use `ccacheStdenv` to drastically improve rebuild time. By default, `ccache` keeps artifacts in `~/.cache/ccache/`.

---

To build Nix itself in this shell:

```
[nix-shell]$./bootstrap.sh
[nix-shell]$./configure $configureFlags --prefix=$(pwd)/outputs/out
[nix-shell]$ make -j $NIX_BUILD_CORES
```

To install it in `$(pwd)/outputs` and test it:

```
[nix-shell]$ make install
[nix-shell]$ make installcheck -j $NIX_BUILD_CORES
[nix-shell]$ nix --version
nix (Nix) 2.12
```

To build a release version of Nix for the current operating system and CPU architecture:

```
$ nix build
```

You can also build Nix for one of the [supported platforms](#).

## 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-clang11StenvPackages
```

---

### Note

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

---

To build Nix itself in this shell:

```
[nix-shell]$./bootstrap.sh
[nix-shell]$./configure $configureFlags --prefix=$(pwd)/outputs/out
[nix-shell]$ make -j $NIX_BUILD_CORES
```

To install it in `$(pwd)/outputs` and test it:

```
[nix-shell]$ make install
[nix-shell]$ make installcheck -j $NIX_BUILD_CORES
[nix-shell]$./outputs/out/bin/nix --version
nix (Nix) 2.12
```

To build a release version of Nix for the current operating system and CPU architecture:

```
$ nix-build
```

You can also build Nix for one of the [supported platforms](#).

## Platforms

Nix can be built for various platforms, as specified in [flake.nix](#):

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

In order to build Nix for a different platform than the one you're currently on, you need a way for your current Nix installation to build code for that platform. Common solutions include [remote builders](#) and [binary format emulation](#) (only supported on NixOS).

Given such a setup, executing the build only requires selecting the respective attribute. For example, to compile for `aarch64-linux`:

```
$ nix-build --attr packages.aarch64-linux.default
```

or for Nix with the `flakes` and `nix-command` experimental features enabled:

```
$ nix build .#packages.aarch64-linux.default
```

Cross-compiled builds are available for ARMv6 (`armv6l-linux`) and ARMv7 (`armv7l-linux`). Add more [system types](#) to `crossSystems` in `flake.nix` to bootstrap Nix on unsupported platforms.

## System type

Nix uses a string with the following format to identify the *system type* or *platform* it runs on:

```
<cpu>-<os>[-<abi>]
```

It is set when Nix is compiled for the given system, and based on the output of `config.guess` ([upstream](#)):

```
<cpu>-<vendor>-<os>[<version>][-<abi>]
```

When Nix is built such that `./configure` is passed any of the `--host`, `--build`, `--target` options, the value is based on the output of `config.sub` ([upstream](#)):

```
<cpu>-<vendor>[-<kernel>]-<os>
```

For historic reasons and backward-compatibility, some CPU and OS identifiers are translated from the GNU Autotools naming convention in `configure.ac` as follows:

| config.guess | Nix    |
|--------------|--------|
| amd64        | x86_64 |
| i*86         | i686   |
| arm6         | arm6l  |
| arm7         | arm7l  |
| linux-gnu*   | linux  |
| linux-musl*  | linux  |

## Compilation environments

Nix can be compiled using multiple environments:

- `stdenv`: default;
- `gccStdenv`: force the use of `gcc` compiler;
- `clangStdenv`: force the use of `clang` compiler;
- `ccacheStdenv`: enable [ccache], a compiler cache to speed up compilation.

To build with one of those environments, you can use

```
$ nix build .#nix-ccacheStdenv
```

for flake-enabled Nix, or

```
$ nix-build --attr nix-ccacheStdenv
```

for classic Nix.

You can use any of the other supported environments in place of `nix-ccacheStdenv`.

## Editor integration

The `clangd` LSP server is installed by default on the `clang`-based `devShell`s. See [supported compilation environments](#) and instructions how to set up a shell [with flakes](#) or in [classic Nix](#).

To use the LSP with your editor, you first need to [set up `clangd`](#) by running:

```
make clean && bear -- make -j$NIX_BUILD_CORES install
```

Configure your editor to use the `clangd` from the shell, either by running it inside the development shell, or by using `nix-direnv` and [the appropriate editor plugin](#).

---

### Note

For some editors (e.g. Visual Studio Code), you may need to install a [special extension](#) for the editor to interact with `clangd`. Some other editors (e.g. Emacs, Vim) need a plugin to support LSP servers in general (e.g. `Isp-mode` for Emacs and `vim-lsp` for vim). Editor-specific setup is typically opinionated, so we will not cover it here in more detail.

---

## Checking links in the manual

The build checks for broken internal links. This happens late in the process, so `nix build` is not suitable for iterating. To build the manual incrementally, run:

```
make html -j $NIX_BUILD_CORES
```

In order to reflect changes to the [Makefile](#), clear all generated files before re-building:

```
rm $(git ls-files doc/manual/ -o | grep -F '.md') && rmdir doc/manual /src/command-ref/new-cli && make html -j $NIX_BUILD_CORES
```

`mdbook-linkcheck` does not implement checking [URI fragments](#) yet.

## .. variable

... provides a base path for links that occur in reusable snippets or other documentation that doesn't have a base path of its own.

If a broken link occurs in a snippet that was inserted into multiple generated files in different directories, use ... to reference the `doc/manual/src` directory.

If the ... literal appears in an error message from the `mdbook-linkcheck` tool, the ... replacement needs to be applied to the generated source file that mentions it. See existing ... logic in the [Makefile](#). Regular markdown files used for the manual have a base path of their own and they can use relative paths instead of ... .

## API documentation

Doxygen API documentation is [available online](#). You can also build and view it yourself:

```
nix build .#hydraJobs.internal-api-docs
xdg-open ./result/share/doc/nix/internal-api/html/index.html
```

or inside a `nix develop` shell by running:

```
make internal-api-html
xdg-open ./outputs/doc/share/doc/nix/internal-api/html/index.html
```

## Coverage analysis

A coverage analysis report is [available online](#). You can build it yourself:

```
nix build .#hydraJobs.coverage
xdg-open ./result/coverage/index.html
```

Metrics about the change in line/function coverage over time are also [available](#).

# Running tests

## Unit-tests

The unit-tests for each Nix library (`libexpr`, `libstore`, etc..) are defined under `tests/unit/{library_name}/tests` using the `googletest` and `rapidcheck` frameworks.

You can run the whole testsuite with `make check`, or the tests for a specific component with `make libfoo-tests_RUN`. Finer-grained filtering is also possible using the `--gtest_filter` command-line option, or the `GTEST_FILTER` environment variable.

## Unit test support libraries

There are headers and code which are not just used to test the library in question, but also downstream libraries. For example, we do [property testing] with the [rapidcheck] library. This requires writing `Arbitrary` "instances", which are used to describe how to generate values of a given type for the sake of running property tests. Because types contain other types, `Arbitrary` "instances" for some type are not just useful for testing that type, but also any other type that contains it. Downstream types frequently contain upstream types, so it is very important that we share arbitrary instances so that downstream libraries' property tests can also use them.

It is important that these testing libraries don't contain any actual tests themselves. On some platforms they would be run as part of every test executable that uses them, which is redundant. On other platforms they wouldn't be run at all.

## Functional tests

The functional tests reside under the `tests/functional` directory and are listed in `tests/functional/local.mk`. Each test is a bash script.

## Running the whole test suite

The whole test suite can be run with:

```
$ make install && make installcheck
ran test tests/functional/foo.sh... [PASS]
ran test tests/functional/bar.sh... [PASS]
...
```

## Grouping tests

Sometimes it is useful to group related tests so they can be easily run together without running the entire test suite. Each test group is in a subdirectory of `tests`. For example, `tests/functional/ca/local.mk` defines a `ca` test group for content-addressed derivation outputs.

That test group can be run like this:

```
$ make ca.test-group -j50
ran test tests/functional/ca/nix-run.sh... [PASS]
ran test tests/functional/ca/import-derivation.sh... [PASS]
...
```

The test group is defined in Make like this:

```
$(test-group-name)-tests := \
 $(d)/test0.sh \
 $(d)/test1.sh \
 ...
install-tests-groups += $(test-group-name)
```

## Running individual tests

Individual tests can be run with `make`:

```
$ make tests/functional/${testName}.sh.test
ran test tests/functional/${testName}.sh... [PASS]
```

or without `make`:

```
$./mk/run-test.sh tests/functional/${testName}.sh
ran test tests/functional/${testName}.sh... [PASS]
```

To see the complete output, one can also run:

```
$./mk/debug-test.sh tests/functional/${testName}.sh
+ foo
output from foo
+ bar
output from bar
...
```

The test script will then be traced with `set -x` and the output displayed as it happens, regardless of whether the test succeeds or fails.

## Debugging failing functional tests

When a functional test fails, it usually does so somewhere in the middle of the script.

To figure out what's wrong, it is convenient to run the test regularly up to the failing `nix` command, and then run that command with a debugger like GDB.

For example, if the script looks like:

```
foo
nix blah blub
bar
```

edit it like so:

```
foo
-nix blah blub
+gdb --args nix blah blub
bar
```

Then, running the test with `./mk/debug-test.sh` will drop you into GDB once the script reaches that point:

```
$./mk/debug-test.sh tests/functional/${testName}.sh
...
+ gdb blah blub
GNU gdb (GDB) 12.1
...
(gdb)
```

One can debug the Nix invocation in all the usual ways. For example, enter `run` to start the Nix invocation.

## Characterization testing

Occasionally, Nix utilizes a technique called [Characterization Testing](#) as part of the functional tests. This technique is to include the exact output/behavior of a former version of Nix in a test in order to check that Nix continues to produce the same behavior going forward.

For example, this technique is used for the language tests, to check both the printed final value if evaluation was successful, and any errors and warnings encountered.

It is frequently useful to regenerate the expected output. To do that, rerun the failed test(s) with `_NIX_TEST_ACCEPT=1`. For example:

```
_NIX_TEST_ACCEPT=1 make tests/functional/lang.sh.test
```

An interesting situation to document is the case when these tests are "overfitted". The language tests are, again, an example of this. The expected successful output of evaluation is supposed to be highly stable – we do not intend to make breaking changes to (the stable parts of) the Nix language. However, the errors and warnings during evaluation (successful or not) are not stable in this way. We are free to change how they are displayed at any time.

It may be surprising that we would test non-normative behavior like diagnostic outputs. Diagnostic outputs are indeed not a stable interface, but they still are important to users. By recording the expected output, the test suite guards against accidental changes, and ensure the *result* (not just the code that implements it) of the diagnostic code paths are under code review. Regressions are caught, and improvements always show up in code review.

To ensure that characterization testing doesn't make it harder to intentionally change these interfaces, there always must be an easy way to regenerate the expected output, as we do with `_NIX_TEST_ACCEPT=1`.

## Integration tests

The integration tests are defined in the Nix flake under the `hydraJobs.tests` attribute. These tests include everything that needs to interact with external services or run Nix in a non-trivial distributed setup. Because these tests are expensive and require more than what the standard github-actions setup provides, they only run on the master branch (on <https://hydra.nixos.org/jobset/nix/master>).

You can run them manually with `nix build .#hydraJobs.tests.{testName}` or `nix-build -A hydraJobs.tests.{testName}`

## Installer tests

After a one-time setup, the Nix repository's GitHub Actions continuous integration (CI) workflow can test the installer each time you push to a branch.

Creating a Cachix cache for your installer tests and adding its authorization token to GitHub enables [two installer-specific jobs in the CI workflow](#):

- The `installer` job generates installers for the platforms below and uploads them to your Cachix cache:
  - `x86_64-linux`
  - `armv6l-linux`
  - `armv7l-linux`
  - `x86_64-darwin`
- The `installer_test` job (which runs on `ubuntu-latest` and `macos-latest`) will try to install Nix with the cached installer and run a trivial Nix command.

## One-time setup

1. Have a GitHub account with a fork of the [Nix repository](#).
2. At [cachix.org](https://cachix.org):
  - Create or log in to an account.
  - Create a Cachix cache using the format `<github-username>-nix-install-tests`.
  - Navigate to the new cache > Settings > Auth Tokens.
  - Generate a new Cachix auth token and copy the generated value.
3. At [github.com](https://github.com):
  - Navigate to your Nix fork > Settings > Secrets > Actions > New repository secret.
  - Name the secret `CACHIX_AUTH_TOKEN`.
  - Paste the copied value of the Cachix cache auth token.

## Working on documentation

### Using the CI-generated installer for manual testing

After the CI run completes, you can check the output to extract the installer URL:

1. Click into the detailed view of the CI run.

2. Click into any `installer_test` run (the URL you're here to extract will be the same in all of them).
3. Click into the `Run cachix/install-nix-action@v...` step and click the detail triangle next to the first log line (it will also be `Run cachix/install-nix-action@v...`)
4. Copy the value of `install_url`
5. To generate an install command, plug this `install_url` and your GitHub username into this template:

```
curl -L <install_url> | sh -s -- --tarball-url-prefix https://<github-username>-nix-install-tests.cachix.org/serve
```

This section describes the notion of *experimental features*, and how it fits into the big picture of the development of Nix.

# What are experimental features?

Experimental features are considered unstable, which means that they can be changed or removed at any time. Users must explicitly enable them by toggling the associated [experimental feature flags](#). This allows accessing unstable functionality without unwittingly relying on it.

Experimental feature flags were first introduced in [Nix 2.4](#). Before that, Nix did have experimental features, but they were not guarded by flags and were merely documented as unstable. This was a source of confusion and controversy.

# When should a new feature be marked experimental?

A change in the Nix codebase should be guarded by an experimental feature flag if it is considered likely to be reverted or adapted in a backwards-incompatible manner after gathering more experience with it in practice.

Examples:

- Changes to the Nix language, such as new built-ins, syntactic or semantic changes, etc.
- Changes to the command-line interface

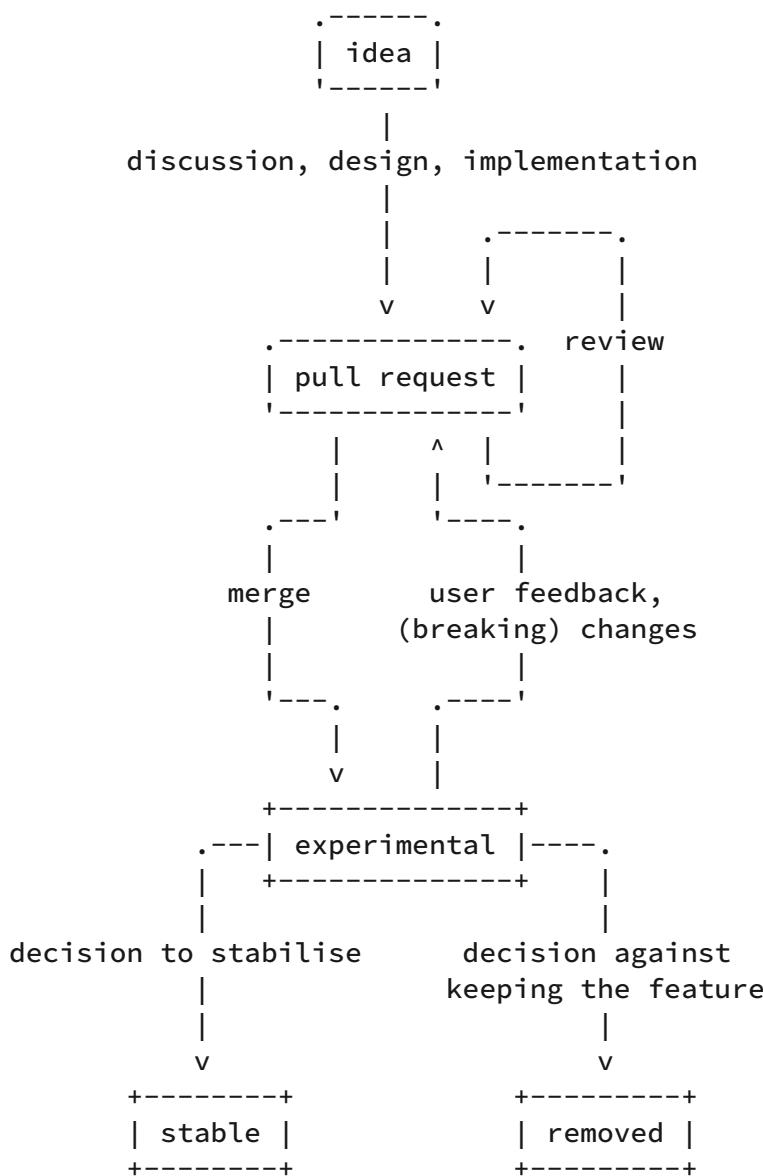
# Lifecycle of an experimental feature

Experimental features have to be treated on a case-by-case basis. However, the standard workflow for an experimental feature is as follows:

- A new feature is implemented in a *pull request*
  - It is guarded by an experimental feature flag that is disabled by default
- The pull request is merged, the *experimental* feature ends up in a release
  - Using the feature requires explicitly enabling it, signifying awareness of the potential risks

- Being experimental, the feature can still be changed arbitrarily
- The feature can be *removed*
  - The associated experimental feature flag is also removed
- The feature can be declared *stable*
  - The associated experimental feature flag is removed
  - There should be enough evidence of users having tried the feature, such as feedback, fixed bugs, demonstrations of how it is put to use
  - Maintainers must feel confident that:
    - The feature is designed and implemented sensibly, that it is fit for purpose
    - Potential interactions are well-understood
    - Stabilising the feature will not incur an outsized maintenance burden in the future

The following diagram illustrates the process:



# Relation to the RFC process

Experimental features and [RFCs](#) both allow approaching substantial changes while minimizing the risk. However they serve different purposes:

- An experimental feature enables developers to iterate on and deliver a new idea without committing to it or requiring a costly long-running fork. It is primarily an issue of *implementation*, targeting Nix developers and early testers.
- The goal of an RFC is to make explicit all the implications of a change: Explain why it is wanted, which new use-cases it enables, which interface changes it requires, etc. It is primarily an issue of *design* and *communication*, targeting the broader community.

This means that experimental features and RFCs are orthogonal mechanisms, and can be used independently or together as needed.

## 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:

- "text hashing" derivation outputs, so we can build .drv files.
- dependencies in derivations on the outputs of derivations that are themselves derivations outputs.

## fetch-closure

Enable the use of the `fetchClosure` built-in function in the Nix language.

## flakes

Enable flakes. See the manual entry for `nix flake` for details.

## impure-derivations

Allow derivations to produce non-fixed outputs by setting the `__impure` derivation attribute to `true`. An impure derivation can have differing outputs each time it is built.

Example:

```
derivation {
 name = "impure";
 builder = /bin/sh;
 __impure = true; # mark this derivation as impure
 args = ["-c" "read -n 10 random < /dev/random; echo $random > $out"];
 system = builtins.currentSystem;
}
```

Each time this derivation is built, it can produce a different output (as the builder outputs random bytes to `$out`). Impure derivations also have access to the network, and only fixed-output or other impure derivations can rely on impure derivations. Finally, an impure derivation cannot also be `content-addressed`.

This is a more explicit alternative to using `builtins.currentTime`.

## nix-command

Enable the new `nix` subcommands. See the manual on `nix` for details.

## no-url-literals

Disallow unquoted URLs as part of the Nix language syntax. The Nix language allows for URL literals, like so:

```
$ nix repl
Welcome to Nix 2.15.0. Type :? for help.

nix-repl> http://foo
"http://foo"
```

But enabling this experimental feature will cause the Nix parser to throw an error when encountering a URL literal:

```
$ nix repl --extra-experimental-features 'no-url-literals'
Welcome to Nix 2.15.0. Type :? for help.

nix-repl> http://foo
error: URL literals are disabled

at «string»:1:1:

1| http://foo
| ^
```

While this is currently an experimental feature, unquoted URLs are being deprecated and their usage is discouraged.

The reason is that, as opposed to path literals, URLs have no special properties that distinguish them from regular strings, URLs containing parameters have to be quoted anyway, and unquoted URLs may confuse external tooling.

## parse-toml-timestamps

Allow parsing of timestamps in `builtins.fromTOML`.

## read-only-local-store

Allow the use of the `read-only` parameter in `local store` URLs.

## recursive-nix

Allow derivation builders to call Nix, and thus build derivations recursively.

Example:

```
with import <nixpkgs> {};

runCommand "foo"
{
 buildInputs = [nix jq];
 NIX_PATH = "nixpkgs=${<nixpkgs>}";
}
```
  hello=$(nix-build -E '(import <nixpkgs> {}).hello.overrideDerivation (args:
{ name = "recursive-hello"; })')

  mkdir -p $out/bin
  ln -s $hello/bin/hello $out/bin/hello
```
``
```

An important restriction on recursive builders is disallowing arbitrary substitutions. For example, running

```
nix-store -r /nix/store/kmwd1hq55akdb9sc7l3finr175dajlby-hello-2.10
```

in the above `runCommand` script would be disallowed, as this could lead to derivations with hidden dependencies or breaking reproducibility by relying on the current state of the Nix store. An exception would be if `/nix/store/kmwd1hq55akdb9sc7l3finr175dajlby-hello-2.10` were already in the build inputs or built by a previous recursive Nix call.

## repl-flake

Allow passing `installables` to `nix repl`, making its interface consistent with the other experimental commands.

# CLI guideline

## Goals

Purpose of this document is to provide a clear direction to **help design delightful command line** experience. This document contains guidelines to follow to ensure a consistent and approachable user experience.

## Overview

`nix` command provides a single entry to a number of sub-commands that help **developers and system administrators** in the life-cycle of a software project. We particularly need to pay special attention to help and assist new users of Nix.

## Naming the COMMANDS

Words matter. Naming is an important part of the usability. Users will be interacting with Nix on a regular basis so we should **name things for ease of understanding**.

We recommend following the [Principle of Least Astonishment](#). This means that you should **never use acronyms or abbreviations** unless they are commonly used in other tools (e.g. `nix init`). And if the command name is too long (> 10-12 characters) then shortening it makes sense (e.g. “prioritization” → “priority”).

Commands should **follow a noun-verb dialogue**. Although noun-verb formatting seems backwards from a speaking perspective (i.e. `nix store copy` vs. `nix copy store`) it allows us to organize commands the same way users think about completing an action (the group first, then the command).

## Naming rules

Rules are there to guide you by limiting your options. But not everything can fit the rules all the time. In those cases document the exceptions in [Appendix 1: Commands naming exceptions](#) and provide reason. The rules want to force a Nix developer to look, not just at the command at hand, but also the command in a full context alongside other `nix` commands.

```
$ nix [<GROUP>] <COMMAND> [<ARGUMENTS>] [<OPTIONS>]
```

- **GROUP**, **COMMAND**, **ARGUMENTS** and **OPTIONS** should be lowercase and in a singular form.
- **GROUP** should be a **NOUN**.
- **COMMAND** should be a **VERB**.
- **ARGUMENTS** and **OPTIONS** are discussed in *Input section*.

## Classification

Some commands are more important, some less. While we want all of our commands to be perfect we can only spend limited amount of time testing and improving them.

This classification tries to separate commands in 3 categories in terms of their importance in regards to the new users. Users who are likely to be impacted the most by bad user experience.

- **Main commands**

Commands used for our main use cases and most likely used by new users. We expect attention to details, such as:

- Proper use of [colors](#), [emojis](#) and [aligning of text](#).
- [Autocomplete](#) of options.
- Show [next possible steps](#).
- Showing some “[tips](#)” when running logs running tasks (eg. building / downloading) in order to teach users interesting bits of Nix ecosystem.
- [Help pages](#) to be as good as we can write them pointing to external documentation and tutorials for more.

Examples of such commands: [nix init](#), [nix develop](#), [nix build](#), [nix run](#), ...

- **Infrequently used commands**

From infrequently used commands we expect less attention to details, but still some:

- Proper use of [colors](#), [emojis](#) and [aligning of text](#).
- [Autocomplete](#) of options.

Examples of such commands: [nix doctor](#), [nix edit](#), [nix eval](#), ...

- **Utility and scripting commands**

Commands that expose certain internal functionality of `nix`, mostly used by other scripts.

- Autocomplete of options.

Examples of such commands: `nix store copy`, `nix hash base16`, `nix store ping`, ...

## Help is essential

Help should be built into your command line so that new users can gradually discover new features when they need them.

## Looking for help

Since there is no standard way how user will look for help we rely on ways help is provided by commonly used tools. As a guide for this we took `git` and whenever in doubt look at it as a preferred direction.

The rules are:

- Help is shown by using `--help` or `help` command (eg `nix --``help` or `nix help`).
- For non-COMMANDs (eg. `nix --``help` and `nix store --``help`) we **show a summary** of most common use cases. Summary is presented on the STDOUT without any use of PAGER.
- For COMMANDs (eg. `nix init --``help` or `nix help init`) we display the man page of that command. By default the PAGER is used (as in `git`).
- At the end of either summary or man page there should be an URL pointing to an online version of more detailed documentation.
- The structure of summaries and man pages should be the same as in `git`.

## Anticipate where help is needed

Even better than requiring the user to search for help is to anticipate and predict when user might need it. Either because the lack of discoverability, typo in the input or simply taking the opportunity to teach the user of interesting - but less visible - details.

## Shell completion

This type of help is most common and almost expected by users. We need to **provide the best shell completion** for `bash`, `zsh` and `fish`.

Completion needs to be **context aware**, this mean when a user types:

```
$ nix build n<TAB>
```

we need to display a list of flakes starting with `n`.

## Wrong input

As we all know we humans make mistakes, all the time. When a typo - intentional or unintentional - is made, we should prompt for closest possible options or point to the documentation which would educate user to not make the same errors. Here are few examples:

In first example we prompt the user for typing wrong command name:

```
$ nix int

Error! Command `int` not found.

Did you mean:
|> nix init
|> nix input
```

Sometimes users will make mistake either because of a typo or simply because of lack of discoverability. Our handling of this cases needs to be context sensitive.

```
$ nix init --template=template#pyton

Error! Template `template#pyton` not found.

Initializing Nix project at `/path/to/here`.
Select a template for your new project:
|> template#python
 template#python-pip
 template#python-poetry
```

## Next steps

It can be invaluable to newcomers to show what a possible next steps and what is the usual

development workflow with Nix. For example:

```
$ nix init --template=template#python
Initializing project `template#python'
 in `/home/USER/dev/new-project'

Next steps
|> nix develop -- to enter development environment
|> nix build -- to build your project
```

## Educate the user

We should take any opportunity to **educate users**, but at the same time we must **be very very careful to not annoy users**. There is a thin line between being helpful and being annoying.

An example of educating users might be to provide *Tips* in places where they are waiting.

```
$ nix build
 Started building my-project 1.2.3
 Downloaded python3.8-poetry 1.2.3 in 5.3 seconds
 Downloaded python3.8-requests 1.2.3 in 5.3 seconds

 Press `v` to increase logs verbosity
 |> `?` to see other options

 Learn something new with every build...
 |> See last logs of a build with `nix log --last` command.

 Evaluated my-project 1.2.3 in 14.43 seconds
 Downloading [12 / 200]
 |> firefox 1.2.3 [#####] 10Mb/s | 2min left
 Building [2 / 20]
 |> glibc 1.2.3 -> buildPhase: <last log line>

```

Now **Learn** part of the output is where you educate users. You should only show it when you know that a build will take some time and not annoy users of the builds that take only few seconds.

Every feature like this should go through an intensive review and testing to collect as much feedback as possible and to fine tune every little detail. If done right this can be an awesome features beginners and advance users will love, but if not done perfectly it will annoy users and leave bad impression.

# Input

Input to a command is provided via **ARGUMENTS** and **OPTIONS**.

**ARGUMENTS** represent a required input for a function. When choosing to use **ARGUMENTS** over **OPTIONS** please be aware of the downsides that come with it:

- User will need to remember the order of **ARGUMENTS**. This is not a problem if there is only one **ARGUMENT**.
- With **OPTIONS** it is possible to provide much better auto completion.
- With **OPTIONS** it is possible to provide much better error message.
- Using **OPTIONS** it will mean there is a little bit more typing.

We don't discourage the use of **ARGUMENTS**, but simply want to make every developer consider the downsides and choose wisely.

## Naming the OPTIONS

The only naming convention - apart from the ones mentioned in Naming the **COMMANDS** section is how flags are named.

Flags are a type of **OPTION** that represent an option that can be turned ON or OFF. We can say **flags are boolean type of **OPTION****.

Here are few examples of flag **OPTIONS**:

- `--colors` vs. `--no-colors` (showing colors in the output)
- `--emojis` vs. `--no-emojis` (showing emojis in the output)

## Prompt when input not provided

For *main commands* (as [per classification](#)) we want command to improve the discoverability of possible input. A new user will most likely not know which **ARGUMENTS** and **OPTIONS** are required or which values are possible for those options.

In case the user does not provide the input or they provide wrong input, rather than show the error, prompt a user with an option to find and select correct input (see examples).

Prompting is of course not required when TTY is not attached to STDIN. This would mean that scripts won't need to handle prompt, but rather handle errors.

A place to use prompt and provide user with interactive select

```
$ nix init
Initializing Nix project at `/path/to/here`.
 Select a template for your new project:
 |> py
 template#python-pip
 template#python-poetry
 [Showing 2 templates from 1345 templates]
```

Another great place to add prompts are **confirmation dialogues for dangerous actions**. For example when adding new substitutor via **OPTIONS** or via **flake.nix** we should prompt - for the first time - and let user review what is going to happen.

```
$ nix build --option substitutors https://cache.example.org

Warning! A security related question needs to be answered.

The following substitutors will be used in `my-project`:
- https://cache.example.org

Do you allow `my-project` to use above mentioned substitutors?
[y/N] |> y
```

# Output

Terminal output can be quite limiting in many ways. Which should force us to think about the experience even more. As with every design the output is a compromise between being terse and being verbose, between showing help to beginners and annoying advance users. For this it is important that we know what are the priorities.

Nix command line should be first and foremost written with beginners in mind. But users won't stay beginners for long and what was once useful might quickly become annoying. There is no golden rule that we can give in this guideline that would make it easier how to draw a line and find best compromise.

What we would encourage is to **build prototypes**, do some **user testing** and collect **feedback**. Then repeat the cycle few times.

First design the *happy path* and only after you iron it out, continue to work on **edge cases** (handling and displaying errors, changes of the output by certain **OPTIONS**, etc...)

## Follow best practices

Needless to say we Nix must be a good citizen and follow best practices in command line.

In short: **STDOUT is for output, STDERR is for (human) messaging.**

STDOUT and STDERR provide a way for you to output messages to the user while also allowing them to redirect content to a file. For example:

```
$ nix build > build.txt

Error! Attribute `bin` missing at (1:94) from string.

1| with import <nixpkgs> { }; (pkgs.runCommandCC or pkgs.runCommand) "shell"
{ buildInputs = [(surge.bin)]; } ""
```

Because this warning is on STDERR, it doesn't end up in the file.

But not everything on STDERR is an error though. For example, you can run `nix build` and collect logs in a file while still seeing the progress.

```
$ nix build > build.txt
Evaluated 1234 files in 1.2 seconds
Downloaded python3.8-poetry 1.2.3 in 5.3 seconds
Downloaded python3.8-requests 1.2.3 in 5.3 seconds

Press `v` to increase logs verbosity
> `?` to see other options
Learn something new with every build...
> See last logs of a build with `nix log --last` command.

Evaluated my-project 1.2.3 in 14.43 seconds
Downloading [12 / 200]
|> firefox 1.2.3 [#####] 10Mb/s | 2min left
Building [2 / 20]
> glibc 1.2.3 -> buildPhase: <last log line>
```

## Errors (WIP)

**TODO:** Once we have implementation for the *happy path* then we will think how to present errors.

## Not only for humans

Terse, machine-readable output formats can also be useful but shouldn't get in the way of making beautiful CLI output. When needed, commands should offer a `--json` flag to allow users to easily parse and script the CLI.

When TTY is not detected on STDOUT we should remove all design elements (no colors, no emojis and using ASCII instead of Unicode symbols). The same should happen when TTY is not detected on STDERR. We should not display progress / status section, but only print warnings and errors.

## Returning future proof JSON

The schema of JSON output should allow for backwards compatible extension. This section explains how to achieve this.

Two definitions are helpful here, because while JSON only defines one "key-value" object type, we use it to cover two use cases:

- **dictionary**: a map from names to value that all have the same type. In C++ this would be a `std::map` with string keys.
- **record**: a fixed set of attributes each with their own type. In C++, this would be represented by a `struct`.

It is best not to mix these use cases, as that may lead to incompatibilities when the schema changes. For example, adding a record field to a dictionary breaks consumers that assume all JSON object fields to have the same meaning and type.

This leads to the following guidelines:

- The top-level (root) value must be a record.

Otherwise, one can not change the structure of a command's output.

- The value of a dictionary item must be a record.

Otherwise, the item type can not be extended.

- List items should be records.

Otherwise, one can not change the structure of the list items.

If the order of the items does not matter, and each item has a unique key that is a string, consider representing the list as a dictionary instead. If the order of the items

needs to be preserved, return a list of records.

- Streaming JSON should return records.

An example of a streaming JSON format is [JSON lines](#), where each line represents a JSON value. These JSON values can be considered top-level values or list items, and they must be records.

## Examples

This is bad, because all keys must be assumed to be store implementations:

```
{
 "local": { ... },
 "remote": { ... },
 "http": { ... }
}
```

This is good, because it is extensible at the root, and is somewhat self-documenting:

```
{
 "storeTypes": { "local": { ... }, ... },
 "pluginSupport": true
}
```

While the dictionary of store types seems like a very complete response at first, a use case may arise that warrants returning additional information. For example, the presence of plugin support may be crucial information for a client to proceed when their desired store type is missing.

The following representation is bad because it is not extensible:

```
{ "outputs": ["out" "bin"] }
```

However, simply converting everything to records is not enough, because the order of outputs must be preserved:

```
{ "outputs": { "bin": {}, "out": {} } }
```

The first item is the default output. Deriving this information from the outputs ordering is not great, but this is how Nix currently happens to work. While it is possible for a JSON parser to preserve the order of fields, we can not rely on this capability to be present in all

JSON libraries.

This representation is extensible and preserves the ordering:

```
{ "outputs": [{ "outputName": "out" }, { "outputName": "bin" }] }
```

## Dialog with the user

CLIs don't always make it clear when an action has taken place. For every action a user performs, your CLI should provide an equal and appropriate reaction, clearly highlighting the what just happened. For example:

```
$ nix build
Downloaded python3.8-poetry 1.2.3 in 5.3 seconds
Downloaded python3.8-requests 1.2.3 in 5.3 seconds
...
Success! You have successfully built my-project.
$
```

Above command clearly states that command successfully completed. And in case of `nix build`, which is a command that might take some time to complete, it is equally important to also show that a command started.

## Text alignment

Text alignment is the number one design element that will present all of the Nix commands as a family and not as separate tools glued together.

The format we should follow is:

```
$ nix COMMAND
 VERB_1 NOUN and other words
 VERB__1 NOUN and other words
 |> Some details
```

Few rules that we can extract from above example:

- Each line should start at least with one space.
- First word should be a VERB and must be aligned to the right.
- Second word should be a NOUN and must be aligned to the left.
- If you can not find a good VERB / NOUN pair, don't worry make it as understandable to the user as possible.

- More details of each line can be provided by `|>` character which is serving as the first word when aligning the text

Don't forget you should also test your terminal output with colors and emojis off (`--no-colors --no-emojis`).

## Dim / Bright

After comparing few terminals with different color schemes we would **recommend to avoid using dimmed text**. The difference from the rest of the text is very little in many terminal and color scheme combinations. Sometimes the difference is not even notable, therefore relying on it wouldn't make much sense.

**The bright text is much better supported** across terminals and color schemes. Most of the time the difference is perceived as if the bright text would be bold.

## Colors

Humans are already conditioned by society to attach certain meaning to certain colors. While the meaning is not universal, a simple collection of colors is used to represent basic emotions.

Colors that can be used in output

- Red = error, danger, stop
- Green = success, good
- Yellow/Orange = proceed with caution, warning, in progress
- Blue/Magenta = stability, calm

While colors are nice, when command line is used by machines (in automation scripts) you want to remove the colors. There should be a global `--no-colors` option that would remove the colors.

## Special (Unicode) characters

Most of the terminal have good support for Unicode characters and you should use them in your output by default. But always have a backup solution that is implemented only with ASCII characters and will be used when `--ascii` option is going to be passed in. Please make sure that you test your output also without Unicode characters

More they showing all the different Unicode characters it is important to **establish common**

**set of characters** that we use for certain situations.

## Emojis

Emojis help channel emotions even better than text, colors and special characters.

We recommend **keeping the set of emojis to a minimum**. This will enable each emoji to stand out more.

As not everybody is happy about emojis we should provide an `--no-emojis` option to disable them. Please make sure that you test your output also without emojis.

## Tables

All commands that are listing certain data can be implemented in some sort of a table. It's important that each row of your output is a single 'entry' of data. Never output table borders. It's noisy and a huge pain for parsing using other tools such as `grep`.

Be mindful of the screen width. Only show a few columns by default with the table header, for more the table can be manipulated by the following options:

- `--no-headers` : Show column headers by default but allow to hide them.
- `--columns` : Comma-separated list of column names to add.
- `--sort` : Allow sorting by column. Allow inverse and multi-column sort as well.

## Interactive output

Interactive output was selected to be able to strike the balance between beginners and advance users. While the default output will target beginners it can, with a few key strokes, be changed into and advance introspection tool.

## Progress

For longer running commands we should provide and overview the progress. This is shown best in `nix build` example:

```
$ nix build
 Started building my-project 1.2.3
Downloaded python3.8-poetry 1.2.3 in 5.3 seconds
Downloaded python3.8-requests 1.2.3 in 5.3 seconds

 Press `v` to increase logs verbosity
> `?` to see other options
Learn something new with every build...
> See last logs of a build with `nix log --last` command.

 Evaluated my-project 1.2.3 in 14.43 seconds
Downloading [12 / 200]
 |> firefox 1.2.3 [#####->] 10Mb/s | 2min left
Building [2 / 20]
> glibc 1.2.3 -> buildPhase: <last log line>
```

## Search

Use a `fzf` like fuzzy search when there are multiple options to choose from.

```
$ nix init
Initializing Nix project at `/path/to/here`.
 Select a template for your new project:
 |> py
 template#python-pip
 template#python-poetry
 [Showing 2 templates from 1345 templates]
```

## Prompt

In some situations we need to prompt the user and inform the user about what is going to happen.

```
$ nix build --option substitutors https://cache.example.org

 Warning! A security related question needs to be answered.

 The following substitutors will be used to in `my-project`:
 - https://cache.example.org
 Do you allow `my-project` to use above mentioned substitutors?
 [y/N] |> y
```

## Verbosity

There are many ways that you can control verbosity.

Verbosity levels are:

- `ERROR` (level 0)
- `WARN` (level 1)
- `NOTICE` (level 2)
- `INFO` (level 3)
- `TALKATIVE` (level 4)
- `CHATTY` (level 5)
- `DEBUG` (level 6)
- `VOMIT` (level 7)

The default level that the command starts is `ERROR`. The simplest way to increase the verbosity by stacking `-v` option (eg: `-vvv == level 3 == INFO`). There are also two shortcuts, `--debug` to run in `DEBUG` verbosity level and `--quiet` to run in `ERROR` verbosity level.

---

## Appendix 1: Commands naming exceptions

`nix init` and `nix repl` are well established

# C++ style guide

Some miscellaneous notes on how we write C++. Formatting we hope to eventually normalize automatically, so this section is free to just discuss higher-level concerns.

## The `*-impl.hh` pattern

Let's start with some background info first. Headers, are supposed to contain declarations, not definitions. This allows us to change a definition without changing the declaration, and have a very small rebuild during development. Templates, however, need to be specialized to use-sites. Absent fancier techniques, templates require that the definition, not just mere declaration, must be available at use-sites in order to make that specialization on the fly as part of compiling those use-sites. Making definitions available like that means putting them in headers, but that is unfortunately means we get all the extra rebuilds we want to avoid by just putting declarations there as described above.

The `*-impl.hh` pattern is a ham-fisted partial solution to this problem. It constitutes:

- Declaring items only in the main `foo.hh`, including templates.
- Putting template definitions in a companion `foo-impl.hh` header.

Most C++ developers would accompany this by having `foo.hh` include `foo-impl.hh`, to ensure any file getting the template declarations also got the template definitions. But we've found not doing this has some benefits and fewer than imagined downsides. The fact remains that headers are rarely as minimal as they could be; there is often code that needs declarations from the headers but not the templates within them. With our pattern where `foo.hh` doesn't include `foo-impl.hh`, that means they can just include `foo.hh`. Code that needs both just includes `foo.hh` and `foo-impl.hh`. This does make linking error possible where something forgets to include `foo-impl.hh` that needs it, but those are build-time only as easy to fix.

# Nix Release Notes

# Release 2.18 (2023-09-20)

- Two new builtin functions, `builtins.parseFlakeRef` and `builtins.flakeRefToString`, have been added. These functions are useful for converting between flake references encoded as attribute sets and URLs.
- `builtins.toJSON` now prints `--show-trace` items for the path in which it finds an evaluation error.
- Error messages regarding malformed input to `nix derivation add` are now clearer and more detailed.
- The `discard-references` feature has been stabilized. This means that the `unsafeDiscardReferences` attribute is no longer guarded by an experimental flag and can be used freely.
- The JSON output for derived paths which are store paths is now a string, not an object with a single `path` field. This only affects `nix-build --json` when "building" non-derivation things like fetched sources, which is a no-op.
- A new builtin `outputOf` has been added. It is part of the `dynamic-derivations` experimental feature.
- Flake follow paths at depths greater than 2 are now handled correctly, preventing "follows a non-existent input" errors.
- `nix-store --query` gained a new type of query: `--valid-derivives`. It returns all `.drv` files in the local store that *can be* used to build the output passed in argument. This is in contrast to `--deriver`, which returns the single `.drv` file that *was actually* used to build the output passed in argument. In case the output was substituted from a binary cache, this `.drv` file may only exist on said binary cache and not locally.

# Release 2.17 (2023-07-24)

- `nix-channel` now supports a `--list-generations` subcommand.
- The function `builtins.fetchClosure` can now fetch input-addressed paths in [pure evaluation mode](#), as those are not impure.
- Nix now allows unprivileged/ `allowed-users` to sign paths. Previously, only `trusted-users` users could sign paths.
- Nested dynamic attributes are now merged correctly by the parser. For example:

```
{
 nested = {
 foo = 1;
 };
 nested = {
 ${"ba" + "r"} = 2;
 };
}
```

This used to silently discard `nested.bar`, but now behaves as one would expect and evaluates to:

```
{ nested = { bar = 2; foo = 1; }; }
```

Note that the feature of merging multiple *full declarations* of attribute sets like `nested` in the example is of questionable value. It allows writing expressions that are very hard to read, for instance when there are many lines of code between two declarations of the same attribute. This has been around for a long time and is therefore supported for backwards compatibility, but should not be relied upon.

Instead, consider using the *nested attribute path* syntax:

```
{
 nested.foo = 1;
 nested.${"ba" + "r"} = 2;
}
```

- Tarball flakes can now redirect to an "immutable" URL that will be recorded in lock

files. This allows the use of "mutable" tarball URLs like `https://example.org/hello/latest.tar.gz` in flakes. See the [tarball fetcher](#) for details.

# Release 2.16 (2023-05-31)

- Speed-up of downloads from binary caches. The number of parallel downloads (also known as substitutions) has been separated from the `--max-jobs` setting. The new setting is called `max-substitution-jobs`. The number of parallel downloads is now set to 16 by default (previously, the default was 1 due to the coupling to build jobs).
- The function `builtins.replaceStrings` is now lazy in the value of its second argument `to`. That is, `to` is only evaluated when its corresponding pattern in `from` is matched in the string `s`.

# Release 2.15 (2023-04-11)

- Commands which take installables on the command line can now read them from the standard input if passed the `--stdin` flag. This is primarily useful when you have a large amount of paths which exceed the OS argument limit.
- The `nix-hash` command now supports Base64 and SRI. Use the flags `--base64` or `--sri` to specify the format of output hash as Base64 or SRI, and `--to-base64` or `--to-sri` to convert a hash to Base64 or SRI format, respectively.

As the choice of hash formats is no longer binary, the `--base16` flag is also added to explicitly specify the Base16 format, which is still the default.

- The special handling of an `installable` with `.drv` suffix being interpreted as all of the given `store derivation`'s output paths is removed, and instead taken as the literal store path that it represents.

The new `^` syntax for store paths introduced in Nix 2.13 allows explicitly referencing output paths of a derivation. Using this is better and more clear than relying on the now-removed `.drv` special handling.

For example,

```
$ nix path-info /nix/store/gzaflydcr6sb3567hap9q6srzx8ggdgg-glibc-
2.33-78 drv
```

now gives info about the derivation itself, while

```
$ nix path-info /nix/store/gzaflydcr6sb3567hap9q6srzx8ggdgg-glibc-
2.33-78 drv^*
```

provides information about each of its outputs.

- The experimental command `nix describe-stores` has been removed.
- Nix stores and their settings are now documented in `nix help-stores`.
- Documentation for operations of `nix-store` and `nix-env` are now available on separate pages of the manual. They include all common options that can be specified and common environment variables that affect these commands.

These pages can be viewed offline with `man` using

- `man nix-store-<operation>` and `man nix-env-<operation>`
  - `nix-store --help --<operation>` and `nix-env --help --<operation>`.
- Nix when used as a client now checks whether the store (the server) trusts the client. (The store always had to check whether it trusts the client, but now the client is informed of the store's decision.) This is useful for scripting interactions with (non-legacy-ssh) remote Nix stores.

`nix store ping` and `nix doctor` now display this information.
  - The new command `nix derivation add` allows adding derivations to the store without involving the Nix language. It exists to round out our collection of basic utility/plumbing commands, and allow for a low barrier-to-entry way of experimenting with alternative front-ends to the Nix Store. It uses the same JSON layout as `nix derivation show`, and is its inverse.
  - `nix show-derivation` has been renamed to `nix derivation show`. This matches `nix derivation add`, and avoids bloating the top-level namespace. The old name is still kept as an alias for compatibility, however.
  - The `nix derivation {add,show}` JSON format now includes the derivation name as a top-level field. This is useful in general, but especially necessary for the `add` direction, as otherwise we would need to pass in the name out of band for certain cases.

# Release 2.14 (2023-02-28)

- A new function `builtins.readFileType` is available. It is similar to `builtins.readDir` but acts on a single file or directory.
- In flakes, the `.outPath` attribute of a flake now always refers to the directory containing the `flake.nix`. This was not the case for when `flake.nix` was in a subdirectory of e.g. a Git repository. The root of the source of a flake in a subdirectory is still available in `.sourceInfo.outPath`.
- In derivations that use structured attributes, you can now use `unsafeDiscardReferences` to disable scanning a given output for runtime dependencies:

```
--structuredAttrs = true;
unsafeDiscardReferences.out = true;
```

This is useful e.g. when generating self-contained filesystem images with their own embedded Nix store: hashes found inside such an image refer to the embedded store and not to the host's Nix store.

This requires the `discard-references` experimental feature.

# Release 2.13 (2023-01-17)

- The `repeat` and `enforce-determinism` options have been removed since they had been broken under many circumstances for a long time.
- You can now use [flake references](#) in the [old command line interface](#), e.g.

```
nix-build flake:nixpkgs -A hello
nix-build -I nixpkgs=flake:github:NixOS/nixpkgs/nixos-22.05 \
'<nixpkgs>' -A hello
NIX_PATH=nixpkgs=flake:nixpkgs nix-build '<nixpkgs>' -A hello
```

- Instead of "antiquotation", the more common term [string interpolation](#) is now used consistently. Historical release notes were not changed.
- Error traces have been reworked to provide detailed explanations and more accurate error locations. A short excerpt of the trace is now shown by default when an error occurs.
- Allow explicitly selecting outputs in a store derivation installable, just like we can do with other sorts of installables. For example,

```
nix build /nix/store/gzaflydcr6sb3567hap9q6srzx8ggdgg-glibc-
2.33-78.drv^dev
```

now works just as

```
nix build nixpkgs#glibc^dev
```

does already.

- On Linux, `nix develop` now sets the [personality](#) for the development shell in the same way as the actual build of the derivation. This makes shells for `i686-linux` derivations work correctly on `x86_64-linux`.
- You can now disable the global flake registry by setting the `flake-registry` configuration option to an empty string. The same can be achieved at runtime with `--flake-registry ""`.

# Release 2.12 (2022-12-06)

- On Linux, Nix can now run builds in a user namespace where they run as root (UID 0) and have 65,536 UIDs available.

This is primarily useful for running containers such as `systemd-nspawn` inside a Nix build. For an example, see [tests/systemd-nspawn/nix](#).

A build can enable this by setting the derivation attribute:

```
requiredSystemFeatures = ["uid-range"];
```

The `uid-range` system feature requires the `auto-allocate-uids` setting to be enabled.

- Nix can now automatically pick UIDs for builds, removing the need to create `nixbld*` user accounts. See [auto-allocate-uids](#).
- On Linux, Nix has experimental support for running builds inside a cgroup. See [use-cgroups](#).
- `<nix/fetchurl.nix>` now accepts an additional argument `impure` which defaults to `false`. If it is set to `true`, the `hash` and `sha256` arguments will be ignored and the resulting derivation will have `__impure` set to `true`, making it an impure derivation.
- If `builtins.readFile` is called on a file with context, then only the parts of the context that appear in the content of the file are retained. This avoids a lot of spurious errors where strings end up having a context just because they are read from a store path ([#7260](#)).
- `nix build --json` now prints some statistics about top-level derivations, such as CPU statistics when cgroups are enabled.

# Release 2.11 (2022-08-24)

- `nix copy` now copies the store paths in parallel as much as possible (again). This doesn't apply for the `daemon` and `ssh-ng` stores which copy everything in one batch to avoid latencies issues.

# Release 2.10 (2022-07-11)

- `nix repl` now takes installables on the command line, unifying the usage with other commands that use `--file` and `--expr`. Primary breaking change is for the common usage of `nix repl '<nixpkgs>'` which can be recovered with `nix repl --file '<nixpkgs>'` or `nix repl --expr 'import <nixpkgs>{}`.

This is currently guarded by the `repl-flake` experimental feature.

- A new function `builtins.traceVerbose` is available. It is similar to `builtins.trace` if the `trace-verbose` setting is set to true, and it is a no-op otherwise.
- `nix search` has a new flag `--exclude` to filter out packages.
- On Linux, if `/nix` doesn't exist and cannot be created and you're not running as root, Nix will automatically use `~/.local/share/nix/root` as a chroot store. This enables non-root users to download the statically linked Nix binary and have it work out of the box, e.g.

```
~/nix run nixpkgs#hello
warning: '/nix' does not exists, so Nix will use '/home/ubuntu/.local/share
/nix/root' as a chroot store
Hello, world!
```

- `flake-registry.json` is now fetched from `channels.nixos.org`.
- Nix can now be built with LTO by passing `--enable-lto` to `configure`. LTO is currently only supported when building with GCC.

# Release 2.9 (2022-05-30)

- Running Nix with the new `--debugger` flag will cause it to start a repl session if an exception is thrown during evaluation, or if `builtins.break` is called. From there you can inspect the values of variables and evaluate Nix expressions. In debug mode, the following new repl commands are available:

|                              |                                                                        |
|------------------------------|------------------------------------------------------------------------|
| <code>:env</code>            | Show env stack                                                         |
| <code>:bt</code>             | Show trace stack                                                       |
| <code>:st</code>             | Show current trace                                                     |
| <code>:st &lt;idx&gt;</code> | Change to another trace in the stack                                   |
| <code>:c</code>              | Go until end of program, exception, or <code>builtins.break()</code> . |
| <code>:s</code>              | Go one step                                                            |

Read more about the debugger [here](#).

- Nix now provides better integration with zsh's `run-help` feature. It is now included in the Nix installation in the form of an autoloadable shell function, `run-help-nix`. It picks up Nix subcommands from the currently typed in command and directs the user to the associated man pages.
- `nix repl` has a new build-and-link (`:bl`) command that builds a derivation while creating GC root symlinks.
- The path produced by `builtins.toFile` is now allowed to be imported or read even with restricted evaluation. Note that this will not work with a read-only store.
- `nix build` has a new `--print-out-paths` flag to print the resulting output paths. This matches the default behaviour of `nix-build`.
- You can now specify which outputs of a derivation `nix` should operate on using the syntax `installable^outputs`, e.g. `nixpkgs#glibc^dev,static` or `nixpkgs#glibc^*`. By default, `nix` will use the outputs specified by the derivation's `meta.outputsToInstall` attribute if it exists, or all outputs otherwise.
- `builtins.fetchTree` (and flake inputs) can now be used to fetch plain files over the `http(s)` and `file` protocols in addition to directory tarballs.

# Release 2.8 (2022-04-19)

- New experimental command: `nix fmt`, which applies a formatter defined by the `formatter.<system>` flake output to the Nix expressions in a flake.
- Various Nix commands can now read expressions from standard input using `--file -`.
- New experimental builtin function `builtins.fetchClosure` that copies a closure from a binary cache at evaluation time and rewrites it to content-addressed form (if it isn't already). Like `builtins.storePath`, this allows importing pre-built store paths; the difference is that it doesn't require the user to configure binary caches and trusted public keys.

This function is only available if you enable the experimental feature `fetch-closure`.

- New experimental feature: *impure derivations*. These are derivations that can produce a different result every time they're built. Here is an example:

```
stdenv.mkDerivation {
 name = "impure";
 __impure = true; # marks this derivation as impure
 buildCommand = "date > $out";
}
```

Running `nix build` twice on this expression will build the derivation twice, producing two different content-addressed store paths. Like fixed-output derivations, impure derivations have access to the network. Only fixed-output derivations and impure derivations can depend on an impure derivation.

- `nix store make-content-addressable` has been renamed to `nix store make-content-addressed`.
- The `nixosModule` flake output attribute has been renamed consistent with the `.default` renames in Nix 2.7.
  - `nixosModule → nixosModules.default`

As before, the old output will continue to work, but `nix flake check` will issue a warning about it.

- `nix run` is now stricter in what it accepts: members of the `apps` flake output are now

required to be apps (as defined in [the manual](#)), and members of `packages` or `legacyPackages` must be derivations (not apps).

# Release 2.7 (2022-03-07)

- Nix will now make some helpful suggestions when you mistype something on the command line. For instance, if you type `nix build nixpkgs#thunderbrd`, it will suggest `thunderbird`.
- A number of "default" flake output attributes have been renamed. These are:
  - `defaultPackage.<system>` → `packages.<system>.default`
  - `defaultApps.<system>` → `apps.<system>.default`
  - `defaultTemplate` → `templates.default`
  - `defaultBundler.<system>` → `bundlers.<system>.default`
  - `overlay` → `overlays.default`
  - `devShell.<system>` → `devShells.<system>.default`

The old flake output attributes still work, but `nix flake check` will warn about them.

- Breaking API change: `nix bundle` now supports bundlers of the form `bundler.<system>.name= derivation: another-derivation;`. This supports additional functionality to inspect evaluation information during bundling. A new `repository` has various bundlers implemented.
- `nix store ping` now reports the version of the remote Nix daemon.
- `nix flake {init,new}` now display information about which files have been created.
- Templates can now define a `welcomeText` attribute, which is printed out by `nix flake {init,new} --template <template>`.

# Release 2.6 (2022-01-24)

- The Nix CLI now searches for a `flake.nix` up until the root of the current Git repository or a filesystem boundary rather than just in the current directory.
- The TOML parser used by `builtins.fromTOML` has been replaced by [a more compliant one](#).
- Added `:st / :show-trace` commands to `nix repl`, which are used to set or toggle display of error traces.
- New builtin function `builtins.zipAttrsWith` with the same functionality as `lib.zipAttrsWith` from Nixpkgs, but much more efficient.
- New command `nix store copy-log` to copy build logs from one store to another.
- The `commit-lockfile-summary` option can be set to a non-empty string to override the commit summary used when committing an updated lockfile. This may be used in conjunction with the `nixConfig` attribute in `flake.nix` to better conform to repository conventions.
- `docker run -ti nixos/nix:master` will place you in the Docker container with the latest version of Nix from the `master` branch.

# Release 2.5 (2021-12-13)

- The garbage collector no longer blocks new builds, so the message `waiting for the big garbage collector lock...` is a thing of the past.
- Binary cache stores now have a setting `compression-level`.
- `nix develop` now has a flag `--unpack` to run `unpackPhase`.
- Lists can now be compared lexicographically using the `<` operator.
- New built-in function: `builtins.groupBy`, with the same functionality as Nixpkgs' `lib.groupBy`, but faster.
- `nix repl` now has a `:log` command.

# Release 2.4 (2021-11-01)

This is the first release in more than two years and is the result of more than 2800 commits from 195 contributors since release 2.3.

## Highlights

- Nix's **error messages** have been improved a lot. For instance, evaluation errors now point out the location of the error:

```
$ nix build
error: undefined variable 'bzip3'

 at /nix/store/449lv242z0zsgwv95a8124xi11sp419f-source
/flake.nix:88:13:

87| [curl
88| bzip3 xz brotli editline
| ^
89| openssl sqlite
```

- The **nix command** has seen a lot of work and is now almost at feature parity with the old command-line interface (the `nix-*` commands). It aims to be **more modern, consistent and pleasant to use** than the old CLI. It is still marked as experimental but its interface should not change much anymore in future releases.
- **Flakes** are a new format to package Nix-based projects in a more discoverable, composable, consistent and reproducible way. A flake is just a repository or tarball containing a file named `flake.nix` that specifies dependencies on other flakes and returns any Nix assets such as packages, Nixpkgs overlays, NixOS modules or CI tests. The new `nix` CLI is primarily based around flakes; for example, a command like `nix run nixpkgs#hello` runs the `hello` application from the `nixpkgs` flake.

Flakes are currently marked as experimental. For an introduction, see [this blog post](#). For detailed information about flake syntax and semantics, see the [nix flake manual page](#).

- Nix's store can now be **content-addressed**, meaning that the hash component of a store path is the hash of the path's contents. Previously Nix could only build **input-addressed** store paths, where the hash is computed from the derivation dependency

graph. Content-addressing allows deduplication, early cutoff in build systems, and unprivileged closure copying. This is still [an experimental feature](#).

- The Nix manual has been converted into Markdown, making it easier to contribute. In addition, every `nix` subcommand now has a manual page, documenting every option.
- A new setting that allows **experimental features** to be enabled selectively. This allows us to merge unstable features into Nix more quickly and do more frequent releases.

## Other features

- There are many new `nix` subcommands:
  - `nix develop` is intended to replace `nix-shell`. It has a number of new features:
    - It automatically sets the output environment variables (such as `$out`) to writable locations (such as `./outputs/out`).
    - It can store the environment in a profile. This is useful for offline work.
    - It can run specific phases directly. For instance, `nix develop --build` runs `buildPhase`.
    - It allows dependencies in the Nix store to be "redirected" to arbitrary directories using the `--redirect` flag. This is useful if you want to hack on a package *and* some of its dependencies at the same time.
  - `nix print-dev-env` prints the environment variables and bash functions defined by a derivation. This is useful for users of other shells than bash (especially with `--json`).
  - `nix shell` was previously named `nix run` and is intended to replace `nix-shell -p`, but without the `stdenv` overhead. It simply starts a shell where some packages have been added to `$PATH`.
  - `nix run` (not to be confused with the old subcommand that has been renamed to `nix shell`) runs an "app", a flake output that specifies a command to run, or an eponymous program from a package. For example, `nix run nixpkgs#hello` runs the `hello` program from the `hello` package in `nixpkgs`.
  - `nix flake` is the container for flake-related operations, such as creating a new flake, querying the contents of a flake or updating flake lock files.

- `nix registry` allows you to query and update the flake registry, which maps identifiers such as `nixpkgs` to concrete flake URLs.
- `nix profile` is intended to replace `nix-env`. Its main advantage is that it keeps track of the provenance of installed packages (e.g. exactly which flake version a package came from). It also has some helpful subcommands:
  - `nix profile history` shows what packages were added, upgraded or removed between each version of a profile.
  - `nix profile diff-closures` shows the changes between the closures of each version of a profile. This allows you to discover the addition or removal of dependencies or size changes.

**Warning:** after a profile has been updated using `nix profile`, it is no longer usable with `nix-env`.

- `nix store diff-closures` shows the differences between the closures of two store paths in terms of the versions and sizes of dependencies in the closures.
- `nix store make-content-addressable` rewrites an arbitrary closure to make it content-addressed. Such paths can be copied into other stores without requiring signatures.
- `nix bundle` uses the `nix-bundle` program to convert a closure into a self-extracting executable.
- Various other replacements for the old CLI, e.g. `nix store gc`, `nix store delete`, `nix store repair`, `nix nar dump-path`, `nix store prefetch-file`, `nix store prefetch-tarball`, `nix key` and `nix daemon`.
- Nix now has an **evaluation cache** for flake outputs. For example, a second invocation of the command `nix run nixpkgs#firefox` will not need to evaluate the `firefox` attribute because it's already in the evaluation cache. This is made possible by the hermetic evaluation model of flakes.
- The new `--offline` flag disables substituters and causes all locally cached tarballs and repositories to be considered up-to-date.
- The new `--refresh` flag causes all locally cached tarballs and repositories to be considered out-of-date.
- Many `nix` subcommands now have a `--json` option to produce machine-readable output.

- `nix repl` has a new `:doc` command to show documentation about builtin functions (e.g. `:doc builtins.map`).
- Binary cache stores now have an option `index-debug-info` to create an index of DWARF debuginfo files for use by `dwarfss`.
- To support flakes, Nix now has an extensible mechanism for fetching source trees. Currently it has the following backends:
  - Git repositories
  - Mercurial repositories
  - GitHub and GitLab repositories (an optimisation for faster fetching than Git)
  - Tarballs
  - Arbitrary directories

The fetcher infrastructure is exposed via flake input specifications and via the `fetchTree` built-in.

- **Languages changes:** the only new language feature is that you can now have antiquotations in paths, e.g. `./${foo}` instead of `./. + foo`.
- **New built-in functions:**
  - `builtins.fetchTree` allows fetching a source tree using any backends supported by the fetcher infrastructure. It subsumes the functionality of existing built-ins like `fetchGit`, `fetchMercurial` and `fetchTarball`.
  - `builtins.getFlake` fetches a flake and returns its output attributes. This function should not be used inside flakes! Use flake inputs instead.
  - `builtins.floor` and `builtins.ceil` round a floating-point number down and up, respectively.
- Experimental support for recursive Nix. This means that Nix derivations can now call Nix to build other derivations. This is not in a stable state yet and not well [documented](#).
- The new experimental feature `no-url-literals` disables URL literals. This helps to implement [RFC 45](#).
- Nix now uses `libarchive` to decompress and unpack tarballs and zip files, so `tar` is no longer required.

- The priority of substituters can now be overridden using the `priority` substituter setting (e.g. `--substituters 'http://cache.nixos.org?priority=100 daemon?priority=10'` ).
- `nix edit` now supports non-derivation attributes, e.g. `nix edit .#nixosConfigurations.bla` .
- The `nix` command now provides command line completion for `bash`, `zsh` and `fish`. Since the support for getting completions is built into `nix`, it's easy to add support for other shells.
- The new `--log-format` flag selects what Nix's output looks like. It defaults to a terse progress indicator. There is a new `internal-json` output format for use by other programs.
- `nix eval` has a new `--apply` flag that applies a function to the evaluation result.
- `nix eval` has a new `--write-to` flag that allows it to write a nested attribute set of string leaves to a corresponding directory tree.
- Memory improvements: many operations that add paths to the store or copy paths between stores now run in constant memory.
- Many `nix` commands now support the flag `--derivation` to operate on a `.drv` file itself instead of its outputs.
- There is a new store called `dummy://` that does not support building or adding paths. This is useful if you want to use the Nix evaluator but don't have a Nix store.
- The `ssh-ng://` store now allows substituting paths on the remote, as `ssh://` already did.
- When auto-calling a function with an ellipsis, all arguments are now passed.
- New `nix-shell` features:
  - It preserves the `PS1` environment variable if `NIX_SHELL_PRESERVE_PROMPT` is set.
  - With `-p`, it passes any `--arg`s as Nixpkgs arguments.
  - Support for structured attributes.
- `nix-prefetch-url` has a new `--executable` flag.
- On `x86_64` systems, `x86_64` microarchitecture levels are mapped to additional system

types (e.g. `x86_64-v1-linux`).

- The new `--eval-store` flag allows you to use a different store for evaluation than for building or storing the build result. This is primarily useful when you want to query whether something exists in a read-only store, such as a binary cache:

```
nix path-info --json --store https://cache.nixos.org \
--eval-store auto nixpkgs#hello
```

(Here `auto` indicates the local store.)

- The Nix daemon has a new low-latency mechanism for copying closures. This is useful when building on remote stores such as `ssh-nc://`.
- Plugins can now register `nix` subcommands.
- The `--indirect` flag to `nix-store --add-root` has become a no-op. `--add-root` will always generate indirect GC roots from now on.

## Incompatible changes

- The `nix` command is now marked as an experimental feature. This means that you need to add

```
experimental-features = nix-command
```

to your `nix.conf` if you want to use it, or pass `--extra-experimental-features nix-command` on the command line.

- The `nix` command no longer has a syntax for referring to packages in a channel. This means that the following no longer works:

```
nix build nixpkgs.hello # Nix 2.3
```

Instead, you can either use the `#` syntax to select a package from a flake, e.g.

```
nix build nixpkgs#hello
```

Or, if you want to use the `nixpkgs` channel in the `NIX_PATH` environment variable:

```
nix build -f '<nixpkgs>' hello
```

- The old `nix run` has been renamed to `nix shell`, while there is a new `nix run` that runs a default command. So instead of

```
nix run nixpkgs.hello -c hello # Nix 2.3
```

you should use

```
nix shell nixpkgs#hello -c hello
```

or just

```
nix run nixpkgs#hello
```

if the command you want to run has the same name as the package.

- It is now an error to modify the `plugin-files` setting via a command-line flag that appears after the first non-flag argument to any command, including a subcommand to `nix`. For example, `nix-instantiate default.nix --plugin-files ""` must now become `nix-instantiate --plugin-files "" default.nix`.
- We no longer release source tarballs. If you want to build from source, please build from the tags in the Git repository.

## Contributors

This release has contributions from Adam Höse, Albert Safin, Alex Kovar, Alex Zero, Alexander Bantyev, Alexandre Esteves, Alyssa Ross, Anatole Lucet, Anders Kaseorg, Andreas Rammhold, Antoine Eiche, Antoine Martin, Arnout Engelen, Arthur Gautier, aszlig, Ben Burdette, Benjamin Hipple, Bernardo Meurer, Björn Gohla, Bjørn Forsman, Bob van der Linden, Brian Leung, Brian McKenna, Brian Wignall, Bruce Toll, Bryan Richter, Calle Rosenquist, Calvin Loncaric, Carlo Nucera, Carlos D'Agostino, Chaz Schlarb, Christian Höppner, Christian Kampka, Chua Hou, Chuck, Cole Helbling, Daiderd Jordan, Dan Callahan, Dani, Daniel Fitzpatrick, Danila Fedorin, Daniël de Kok, Danny Bautista, DavHau, David McFarland, Dima, Domen Kožar, Dominik Schrempf, Dominique Martinet, dramforever, Dustin DeWeese, edef, Eelco Dolstra, Ellie Hermaszewska, Emilio Karakey, Emily, Eric Culp, Ersin Akinci, Fabian Möller, Farid Zakaria, Federico Pellegrin, Finn Behrens, Florian Franzen, Félix Baylac-Jacqué, Gabriella Gonzalez, Geoff Reedy, Georges Dubus, Graham Christensen, Greg Hale, Greg Price, Gregor Kleen, Gregory Hale, Griffin Smith, Guillaume Bouchard,

Harald van Dijk, illustris, Ivan Zvonimir Horvat, Jade, Jake Waksbaum, jakobrs, James Ottaway, Jan Tojnar, Janne Heß, Jaroslavas Pocepko, Jarrett Keifer, Jeremy Schlatter, Joachim Breitner, Joe Pea, John Ericson, Jonathan Ringer, Josef Kemetmüller, Joseph Lucas, Jude Taylor, Julian Stecklina, Julien Tanguy, Jörg Thalheim, Kai Wohlfahrt, keke, Keshav Kini, Kevin Quick, Kevin Stock, Kjetil Orbekk, Krzysztof Gogolewski, kvtb, Lars Mühlmel, Leonhard Markert, Lily Ballard, Linus Heckemann, Lorenzo Manacorda, Lucas Desgouilles, Lucas Franceschino, Lucas Hoffmann, Luke Granger-Brown, Madeline Haraj, Marwan Aljubeh, Mat Marini, Mateusz Piotrowski, Matthew Bauer, Matthew Kenigsberg, Mauricio Scheffer, Maximilian Bosch, Michael Adler, Michael Bishop, Michael Fellinger, Michael Forney, Michael Reilly, mlatus, Mykola Orliuk, Nathan van Doorn, Naïm Favier, ng0, Nick Van den Broeck, Nicolas Stig124 Formichella, Niels Egberts, Niklas Hambüchen, Nikola Knezevic, oxalica, p01arst0rm, Pamplemousse, Patrick Hilhorst, Paul Opiyo, Pavol Rusnak, Peter Kolloch, Philipp Bartsch, Philipp Middendorf, Piotr Szubiakowski, Profpatsch, Puck Meerburg, Ricardo M. Correia, Rickard Nilsson, Robert Hensing, Robin Gloster, Rodrigo, Rok Garbas, Ronnie Ebrin, Rovanion Luckey, Ryan Burns, Ryan Mulligan, Ryne Everett, Sam Doshi, Sam Lidder, Samir Talwar, Samuel Dionne-Riel, Sebastian Ullrich, Sergei Trofimovich, Sevan Janiyan, Shao Cheng, Shea Levy, Silvan Mosberger, Stefan Frijters, Stefan Jaax, sternenseemann, Steven Shaw, Stéphan Kochen, SuperSandro2000, Suraj Barkale, Taeer Bar-Yam, Thomas Churchman, Théophane Hufschmitt, Timothy DeHerrera, Timothy Klim, Tobias Möst, Tobias Pflug, Tom Bereknyei, Travis A. Everett, Ujjwal Jain, Vladimír Čunát, Wil Taylor, Will Dietz, Yaroslav Bolyukin, Yestin L. Harrison, Yi, Yorick van Pelt, Yuriy Taraday and zimbatm.

# Release 2.3 (2019-09-04)

This is primarily a bug fix release. However, it makes some incompatible changes:

- Nix now uses BSD file locks instead of POSIX file locks. Because of this, you should not use Nix 2.3 and previous releases at the same time on a Nix store.

It also has the following changes:

- `builtins.fetchGit`'s `ref` argument now allows specifying an absolute remote ref. Nix will automatically prefix `ref` with `refs/heads` only if `ref` doesn't already begin with `refs/`.
- The installer now enables sandboxing by default on Linux when the system has the necessary kernel support.
- The `max-jobs` setting now defaults to 1.
- New builtin functions: `builtins.isPath`, `builtins.hashFile`.
- The `nix` command has a new `--print-build-logs` (`-L`) flag to print build log output to stderr, rather than showing the last log line in the progress bar. To distinguish between concurrent builds, log lines are prefixed by the name of the package.
- Builds are now executed in a pseudo-terminal, and the `TERM` environment variable is set to `xterm-256color`. This allows many programs (e.g. `gcc`, `clang`, `cmake`) to print colorized log output.
- Add `--no-net` convenience flag. This flag disables substituters; sets the `tarball-ttl` setting to infinity (ensuring that any previously downloaded files are considered current); and disables retrying downloads and sets the connection timeout to the minimum. This flag is enabled automatically if there are no configured non-loopback network interfaces.
- Add a `post-build-hook` setting to run a program after a build has succeeded.
- Add a `trace-function-calls` setting to log the duration of Nix function calls to stderr.

# Release 2.2 (2019-01-11)

This is primarily a bug fix release. It also has the following changes:

- In derivations that use structured attributes (i.e. that specify set the `__structuredAttrs` attribute to `true` to cause all attributes to be passed to the builder in JSON format), you can now specify closure checks per output, e.g.:

```
outputChecks."out" = {
 # The closure of 'out' must not be larger than 256 MiB.
 maxClosureSize = 256 * 1024 * 1024;

 # It must not refer to C compiler or to the 'dev' output.
 disallowedRequisites = [stdenv.cc "dev"];
};

outputChecks."dev" = {
 # The 'dev' output must not be larger than 128 KiB.
 maxSize = 128 * 1024;
};
```

- The derivation attribute `requiredSystemFeatures` is now enforced for local builds, and not just to route builds to remote builders. The supported features of a machine can be specified through the configuration setting `system-features`.

By default, `system-features` includes `kvm` if `/dev/kvm` exists. For compatibility, it also includes the pseudo-features `nixos-test`, `benchmark` and `big-parallel` which are used by Nixpkgs to route builds to particular Hydra build machines.

- Sandbox builds are now enabled by default on Linux.
- The new command `nix doctor` shows potential issues with your Nix installation.
- The `fetchGit` builtin function now uses a caching scheme that puts different remote repositories in distinct local repositories, rather than a single shared repository. This may require more disk space but is faster.
- The `dirOf` builtin function now works on relative paths.
- Nix now supports [SRI hashes](#), allowing the hash algorithm and hash to be specified in a single string. For example, you can write:

```
import <nix/fetchurl.nix> {
 url = https://nixos.org/releases/nix/nix-2.1.3/nix-2.1.3.tar.xz;
 hash = "sha256-XSLa0FjVyADWWhFfkZ2iKTjFDda6mMXjoYMXLRSYQKQ=";
};
```

instead of

```
import <nix/fetchurl.nix> {
 url = https://nixos.org/releases/nix/nix-2.1.3/nix-2.1.3.tar.xz;
 sha256 =
"5d22dad058d5c800d65a115f919da22938c50dd6ba98c5e3a183172d149840a4";
};
```

In fixed-output derivations, the `outputHashAlgo` attribute is no longer mandatory if `outputHash` specifies the hash.

`nix hash-file` and `nix hash-path` now print hashes in SRI format by default. They also use SHA-256 by default instead of SHA-512 because that's what we use most of the time in Nixpkgs.

- Integers are now 64 bits on all platforms.
- The evaluator now prints profiling statistics (enabled via the `NIX_SHOW_STATS` and `NIX_COUNT_CALLS` environment variables) in JSON format.
- The option `--xml` in `nix-store --query` has been removed. Instead, there now is an option `--graphml` to output the dependency graph in GraphML format.
- All `nix-*` commands are now symlinks to `nix`. This saves a bit of disk space.
- `nix repl` now uses `libeditline` or `libreadline`.

# Release 2.1 (2018-09-02)

This is primarily a bug fix release. It also reduces memory consumption in certain situations. In addition, it has the following new features:

- The Nix installer will no longer default to the Multi-User installation for macOS. You can still instruct the installer to run in multi-user mode.
- The Nix installer now supports performing a Multi-User installation for Linux computers which are running systemd. You can select a Multi-User installation by passing the `--daemon` flag to the installer: `sh <(curl -L https://nixos.org/nix/install) --daemon`.

The multi-user installer cannot handle systems with SELinux. If your system has SELinux enabled, you can force the installer to run in single-user mode.

- New builtin functions: `builtins.bitAnd`, `builtins.bitOr`, `builtins.bitXor`, `builtins.fromTOML`, `builtins.concatMap`, `builtins.mapAttrs`.
- The S3 binary cache store now supports uploading NARs larger than 5 GiB.
- The S3 binary cache store now supports uploading to S3-compatible services with the `endpoint` option.
- The flag `--fallback` is no longer required to recover from disappeared NARs in binary caches.
- `nix-daemon` now respects `--store`.
- `nix run` now respects `nix-support/propagated-user-env-packages`.

This release has contributions from Adrien Devresse, Aleksandr Pashkov, Alexandre Esteves, Amine Chikhaoui, Andrew Dunham, Asad Saeeduddin, aszlig, Ben Challenor, Ben Gamari, Benjamin Hipple, Bogdan Seniuc, Corey O'Connor, Daiderd Jordan, Daniel Peebles, Daniel Poelzleithner, Danylo Hlynskyi, Dmitry Kalinkin, Domen Kožar, Doug Beardsley, Eelco Dolstra, Erik Arvstedt, Félix Baylac-Jacqué, Gleb Peregud, Graham Christensen, Guillaume Maudoux, Ivan Kozik, John Arnold, Justin Humm, Linus Heckemann, Lorenzo Manacorda, Matthew Justin Bauer, Matthew O'Gorman, Maximilian Bosch, Michael Bishop, Michael Fiano, Michael Mercier, Michael Raskin, Michael Weiss, Nicolas Dudebout, Peter Simons, Ryan Trinkle, Samuel Dionne-Riel, Sean Seefried, Shea Levy, Symphorien Gibol, Tim Engler, Tim Sears, Tuomas Tynkkynen, volth, Will Dietz, Yorick van Pelt and zimbatm.

# Release 2.0 (2018-02-22)

The following incompatible changes have been made:

- The manifest-based substituter mechanism ([download-using-manifests](#)) has been [removed](#). It has been superseded by the binary cache substituter mechanism since several years. As a result, the following programs have been removed:
  - `nix-pull`
  - `nix-generate-patches`
  - `bsdiff`
  - `bspatch`
- The “copy from other stores” substituter mechanism ([copy-from-other-stores](#) and the `NIX_OTHER_STORES` environment variable) has been removed. It was primarily used by the NixOS installer to copy available paths from the installation medium. The replacement is to use a chroot store as a substituter (e.g. `--substituters /mnt`), or to build into a chroot store (e.g. `--store /mnt --substituters /`).
- The command `nix-push` has been removed as part of the effort to eliminate Nix's dependency on Perl. You can use `nix copy` instead, e.g. `nix copy --to file:///tmp /my-binary-cache paths...`
- The “nested” log output feature ([--log-type pretty](#)) has been removed. As a result, `nix-log2xml` was also removed.
- OpenSSL-based signing has been [removed](#). This feature was never well-supported. A better alternative is provided by the `secret-key-files` and `trusted-public-keys` options.
- Failed build caching has been [removed](#). This feature was introduced to support the Hydra continuous build system, but Hydra no longer uses it.
- `nix-mode.el` has been removed from Nix. It is now [a separate repository](#) and can be installed through the MELPA package repository.

This release has the following new features:

- It introduces a new command named `nix`, which is intended to eventually replace all `nix-*` commands with a more consistent and better designed user interface. It

currently provides replacements for some (but not all) of the functionality provided by `nix-store`, `nix-build`, `nix-shell -p`, `nix-env -qa`, `nix-instantiate --eval`, `nix-push` and `nix-copy-closure`. It has the following major features:

- Unlike the legacy commands, it has a consistent way to refer to packages and package-like arguments (like store paths). For example, the following commands all copy the GNU Hello package to a remote machine:

```
nix copy --to ssh://machine nixpkgs.hello
```

```
nix copy --to ssh://machine /nix/store
/0i2jd68mp5g6h2sa5k9c85rb80sn8hi9-hello-2.10
```

```
nix copy --to ssh://machine '(with import <nixpkgs> {}; hello)'
```

By contrast, `nix-copy-closure` only accepted store paths as arguments.

- It is self-documenting: `--help` shows all available command-line arguments. If `--help` is given after a subcommand, it shows examples for that subcommand. `nix --help-config` shows all configuration options.
- It is much less verbose. By default, it displays a single-line progress indicator that shows how many packages are left to be built or downloaded, and (if there are running builds) the most recent line of builder output. If a build fails, it shows the last few lines of builder output. The full build log can be retrieved using `nix log`.
- It provides all `nix.conf` configuration options as command line flags. For example, instead of `--option http-connections 100` you can write `--http-connections 100`. Boolean options can be written as `--foo` or `--no-foo` (e.g. `--no-auto-optimise-store`).
- Many subcommands have a `--json` flag to write results to stdout in JSON format.

---

## Warning

Please note that the `nix` command is a work in progress and the interface is subject to change.

---

It provides the following high-level (“porcelain”) subcommands:

- `nix build` is a replacement for `nix-build`.

- `nix run` executes a command in an environment in which the specified packages are available. It is (roughly) a replacement for `nix-shell -p`. Unlike that command, it does not execute the command in a shell, and has a flag (`-c`) that specifies the unquoted command line to be executed.

It is particularly useful in conjunction with chroot stores, allowing Linux users who do not have permission to install Nix in `/nix/store` to still use binary substitutes that assume `/nix/store`. For example,

```
nix run --store ~/my-nix nixpkgs.hello -c hello --greeting 'Hi
everybody!'
```

downloads (or if no substitutes are available, builds) the GNU Hello package into `~/my-nix/nix/store`, then runs `hello` in a mount namespace where `~/my-nix/nix/store` is mounted onto `/nix/store`.

- `nix search` replaces `nix-env -qa`. It searches the available packages for occurrences of a search string in the attribute name, package name or description. Unlike `nix-env -qa`, it has a cache to speed up subsequent searches.
- `nix copy` copies paths between arbitrary Nix stores, generalising `nix-copy-closure` and `nix-push`.
- `nix repl` replaces the external program `nix-repl`. It provides an interactive environment for evaluating and building Nix expressions. Note that it uses `linenoise-ng` instead of GNU Readline.
- `nix upgrade-nix` upgrades Nix to the latest stable version. This requires that Nix is installed in a profile. (Thus it won't work on NixOS, or if it's installed outside of the Nix store.)
- `nix verify` checks whether store paths are unmodified and/or "trusted" (see below). It replaces `nix-store --verify` and `nix-store --verify-path`.
- `nix log` shows the build log of a package or path. If the build log is not available locally, it will try to obtain it from the configured substituters (such as `cache.nixos.org`, which now provides build logs).
- `nix edit` opens the source code of a package in your editor.
- `nix eval` replaces `nix-instantiate --eval`.

- `nix why-depends` shows why one store path has another in its closure. This is primarily useful to finding the causes of closure bloat. For example,

```
nix why-depends nixpkgs.vlc nixpkgs.libdrm.dev
```

shows a chain of files and fragments of file contents that cause the VLC package to have the “dev” output of `libdrm` in its closure — an undesirable situation.

- `nix path-info` shows information about store paths, replacing `nix-store -q`. A useful feature is the option `--closure-size` (`-S`). For example, the following command show the closure sizes of every path in the current NixOS system closure, sorted by size:

```
nix path-info -rS /run/current-system | sort -nk2
```

- `nix optimise-store` replaces `nix-store --optimise`. The main difference is that it has a progress indicator.

A number of low-level (“plumbing”) commands are also available:

- `nix ls-store` and `nix ls-nar` list the contents of a store path or NAR file. The former is primarily useful in conjunction with remote stores, e.g.

```
nix ls-store --store https://cache.nixos.org/ -lR /nix/store
/0i2jd68mp5g6h2sa5k9c85rb80sn8hi9-hello-2.10
```

lists the contents of path in a binary cache.

- `nix cat-store` and `nix cat-nar` allow extracting a file from a store path or NAR file.
- `nix dump-path` writes the contents of a store path to stdout in NAR format. This replaces `nix-store --dump`.
- `nix show-derivation` displays a store derivation in JSON format. This is an alternative to `pp-aterm`.
- `nix add-to-store` replaces `nix-store --add`.
- `nix sign-paths` signs store paths.
- `nix copy-sigs` copies signatures from one store to another.

- `nix show-config` shows all configuration options and their current values.
- The store abstraction that Nix has had for a long time to support store access via the Nix daemon has been extended significantly. In particular, substituters (which used to be external programs such as `download-from-binary-cache`) are now subclasses of the abstract `Store` class. This allows many Nix commands to operate on such store types. For example, `nix path-info` shows information about paths in your local Nix store, while `nix path-info --store https://cache.nixos.org/` shows information about paths in the specified binary cache. Similarly, `nix-copy-closure`, `nix-push` and substitution are all instances of the general notion of copying paths between different kinds of Nix stores.

Stores are specified using an URI-like syntax, e.g. <https://cache.nixos.org/> or <ssh://machine>. The following store types are supported:

- `LocalStore` (stori URI `local` or an absolute path) and the misnamed `RemoteStore` (`daemon`) provide access to a local Nix store, the latter via the Nix daemon. You can use `auto` or the empty string to auto-select a local or daemon store depending on whether you have write permission to the Nix store. It is no longer necessary to set the `NIX_REMOTE` environment variable to use the Nix daemon.

As noted above, `LocalStore` now supports chroot builds, allowing the “physical” location of the Nix store (e.g. `/home/alice/nix/store`) to differ from its “logical” location (typically `/nix/store`). This allows non-root users to use Nix while still getting the benefits from prebuilt binaries from [cache.nixos.org](https://cache.nixos.org).

- `BinaryCacheStore` is the abstract superclass of all binary cache stores. It supports writing build logs and NAR content listings in JSON format.
- `HttpBinaryCacheStore` (`http://`, `https://`) supports binary caches via HTTP or HTTPS. If the server supports `PUT` requests, it supports uploading store paths via commands such as `nix copy`.
- `LocalBinaryCacheStore` (`file://`) supports binary caches in the local filesystem.
- `S3BinaryCacheStore` (`s3://`) supports binary caches stored in Amazon S3, if enabled at compile time.
- `LegacySSHStore` (`ssh://`) is used to implement remote builds and `nix-copy-closure`.

- `SSHStore` (`ssh-ng://`) supports arbitrary Nix operations on a remote machine via the same protocol used by `nix-daemon`.
- Security has been improved in various ways:
  - Nix now stores signatures for local store paths. When paths are copied between stores (e.g., copied from a binary cache to a local store), signatures are propagated.

Locally-built paths are signed automatically using the secret keys specified by the `secret-key-files` store option. Secret/public key pairs can be generated using `nix-store --generate-binary-cache-key`.

In addition, locally-built store paths are marked as “ultimately trusted”, but this bit is not propagated when paths are copied between stores.

    - Content-addressable store paths no longer require signatures — they can be imported into a store by unprivileged users even if they lack signatures.
    - The command `nix verify` checks whether the specified paths are trusted, i.e., have a certain number of trusted signatures, are ultimately trusted, or are content-addressed.
    - Substitutions from binary caches `now` require signatures by default. This was already the case on NixOS.
    - In Linux sandbox builds, we `now` use `/build` instead of `/tmp` as the temporary build directory. This fixes potential security problems when a build accidentally stores its `TMPDIR` in some security-sensitive place, such as an RPATH.
- *Pure evaluation mode.* With the `--pure-eval` flag, Nix enables a variant of the existing restricted evaluation mode that forbids access to anything that could cause different evaluations of the same command line arguments to produce a different result. This includes builtin functions such as `builtins.getenv`, but more importantly, *all* filesystem or network access unless a content hash or commit hash is specified. For example, calls to `builtins.fetchGit` are only allowed if a `rev` attribute is specified.

The goal of this feature is to enable true reproducibility and traceability of builds (including NixOS system configurations) at the evaluation level. For example, in the future, `nixos-rebuild` might build configurations from a Nix expression in a Git repository in pure mode. That expression might fetch other repositories such as Nixpkgs via `builtins.fetchGit`. The commit hash of the top-level repository then uniquely identifies a running system, and, in conjunction with that repository, allows it

to be reproduced or modified.

- There are several new features to support binary reproducibility (i.e. to help ensure that multiple builds of the same derivation produce exactly the same output). When `enforce-determinism` is set to `false`, it's `no longer` a fatal error if build rounds produce different output. Also, a hook named `diff-hook` is `provided` to allow you to run tools such as `diffoscope` when build rounds produce different output.
- Configuring remote builds is a lot easier now. Provided you are not using the Nix daemon, you can now just specify a remote build machine on the command line, e.g. `--option builders 'ssh://my-mac x86_64-darwin'`. The environment variable `NIX_BUILD_HOOK` has been removed and is no longer needed. The environment variable `NIX_REMOTE_SYSTEMS` is still supported for compatibility, but it is also possible to specify builders in `nix.conf` by setting the option `builders = @path`.
- If a fixed-output derivation produces a result with an incorrect hash, the output path is moved to the location corresponding to the actual hash and registered as valid. Thus, a subsequent build of the fixed-output derivation with the correct hash is unnecessary.
- `nix-shell` now sets the `IN_NIX_SHELL` environment variable during evaluation and in the shell itself. This can be used to perform different actions depending on whether you're in a Nix shell or in a regular build. Nixpkgs provides `lib.inNixShell` to check this variable during evaluation.
- `NIX_PATH` is now lazy, so URIs in the path are only downloaded if they are needed for evaluation.
- You can now use `channel:` as a short-hand for <https://nixos.org/channels//nixexprs.tar.xz>. For example, `nix-build channel:nixos-15.09 -A hello` will build the GNU Hello package from the `nixos-15.09` channel. In the future, this may use Git to fetch updates more efficiently.
- When `--no-build-output` is given, the last 10 lines of the build log will be shown if a build fails.
- Networking has been improved:
  - HTTP/2 is now supported. This makes binary cache lookups `much more efficient`.
  - We now retry downloads on many HTTP errors, making binary caches substituters more resilient to temporary failures.
  - HTTP credentials can now be configured via the standard `netrc` mechanism.

- If S3 support is enabled at compile time, `s3://` URLs are supported in all places where Nix allows URIs.
- Brotli compression is now supported. In particular, [cache.nixos.org](#) build logs are now compressed using Brotli.
- `nix-env` now ignores packages with bad derivation names (in particular those starting with a digit or containing a dot).
- Many configuration options have been renamed, either because they were unnecessarily verbose (e.g. `build-use-sandbox` is now just `sandbox`) or to reflect generalised behaviour (e.g. `binary-caches` is now `substituters` because it allows arbitrary store URIs). The old names are still supported for compatibility.
- The `max-jobs` option can now be set to `auto` to use the number of CPUs in the system.
- Hashes can now be specified in base-64 format, in addition to base-16 and the non-standard base-32.
- `nix-shell` now uses `bashInteractive` from Nixpkgs, rather than the `bash` command that happens to be in the caller's `PATH`. This is especially important on macOS where the `bash` provided by the system is seriously outdated and cannot execute `stdenv`'s setup script.
- Nix can now automatically trigger a garbage collection if free disk space drops below a certain level during a build. This is configured using the `min-free` and `max-free` options.
- `nix-store -q --roots` and `nix-store --gc --print-roots` now show temporary and in-memory roots.
- Nix can now be extended with plugins. See the documentation of the `plugin-files` option for more details.

The Nix language has the following new features:

- It supports floating point numbers. They are based on the C++ `float` type and are supported by the existing numerical operators. Export and import to and from JSON and XML works, too.
- Derivation attributes can now reference the outputs of the derivation using the `placeholder` builtin function. For example, the attribute

```
configureFlags = "--prefix=${placeholder \"out\"} --includedir=${placeholder \"dev\"}";
```

will cause the `configureFlags` environment variable to contain the actual store paths corresponding to the `out` and `dev` outputs.

The following builtin functions are new or extended:

- `builtins.fetchGit` allows Git repositories to be fetched at evaluation time. Thus it differs from the `fetchgit` function in Nixpkgs, which fetches at build time and cannot be used to fetch Nix expressions during evaluation. A typical use case is to import external NixOS modules from your configuration, e.g.

```
imports = [(builtins.fetchGit https://github.com/edolstra/dwarfes + "/module.nix")];
```

- Similarly, `builtins.fetchMercurial` allows you to fetch Mercurial repositories.
- `builtins.path` generalises `builtins.filterSource` and path literals (e.g. `./foo`). It allows specifying a store path name that differs from the source path name (e.g. `builtins.path { path = ./foo; name = "bar"; }`) and also supports filtering out unwanted files.
- `builtins.fetchurl` and `builtins.fetchTarball` now support `sha256` and `name` attributes.
- `builtins.split` splits a string using a POSIX extended regular expression as the separator.
- `builtins.partition` partitions the elements of a list into two lists, depending on a Boolean predicate.
- `<nix/fetchurl.nix>` now uses the content-addressable tarball cache at <http://tarballs.nixos.org/>, just like `fetchurl` in Nixpkgs.  
(f2682e6e18a76ecfb8a12c17e3a0ca15c084197)
- In restricted and pure evaluation mode, builtin functions that download from the network (such as `fetchGit`) are permitted to fetch underneath a list of URI prefixes specified in the option `allowed-uris`.

The Nix build environment has the following changes:

- Values such as Booleans, integers, (nested) lists and attribute sets can [now](#) be passed

to builders in a non-lossy way. If the special attribute `__structuredAttrs` is set to `true`, the other derivation attributes are serialised in JSON format and made available to the builder via the file `.attrs.json` in the builder's temporary directory. This obviates the need for `passAsFile` since JSON files have no size restrictions, unlike process environments.

As a convenience to Bash builders, Nix writes a script named `.attrs.sh` to the builder's directory that initialises shell variables corresponding to all attributes that are representable in Bash. This includes non-nested (associative) arrays. For example, the attribute `hardening.format = true` ends up as the Bash associative array element  `${hardening[format]}` .

- Builders can now communicate what build phase they are in by writing messages to the file descriptor specified in `NIX_LOG_FD`. The current phase is shown by the `nix` progress indicator.
- In Linux sandbox builds, we now provide a default `/bin/sh` (namely `ash` from BusyBox).
- In structured attribute mode, `exportReferencesGraph` exports extended information about closures in JSON format. In particular, it includes the sizes and hashes of paths. This is primarily useful for NixOS image builders.
- Builds are now killed as soon as Nix receives EOF on the builder's stdout or stderr. This fixes a bug that allowed builds to hang Nix indefinitely, regardless of timeouts.
- The `sandbox-paths` configuration option can now specify optional paths by appending a `?`, e.g. `/dev/nvidiactl?` will bind-mount `/dev/nvidiactl` only if it exists.
- On Linux, builds are now executed in a user namespace with UID 1000 and GID 100.

A number of significant internal changes were made:

- Nix no longer depends on Perl and all Perl components have been rewritten in C++ or removed. The Perl bindings that used to be part of Nix have been moved to a separate package, `nix-perl`.
- All `Store` classes are now thread-safe. `RemoteStore` supports multiple concurrent connections to the daemon. This is primarily useful in multi-threaded programs such as `hydra-queue-runner`.

This release has contributions from Adrien Devresse, Alexander Ried, Alex Cruice, Alexey Shmalko, AmineChikhaoui, Andy Wingo, Aneesh Agrawal, Anthony Cowley, Armijn Hemel, aszlig, Ben Gamari, Benjamin Hipple, Benjamin Staffin, Benno Fünfstück, Bjørn Forsman,

Brian McKenna, Charles Strahan, Chase Adams, Chris Martin, Christian Theune, Chris Warburton, Daiderd Jordan, Dan Connolly, Daniel Peebles, Dan Peebles, davidak, David McFarland, Dmitry Kalinkin, Domen Kožar, Eelco Dolstra, Emery Hemingway, Eric Litak, Eric Wolf, Fabian Schmitthenner, Frederik Rietdijk, Gabriel Gonzalez, Giorgio Gallo, Graham Christensen, Guillaume Maudoux, Harmen, lavael, James Broadhead, James Earl Douglas, Janus Troelsen, Jeremy Shaw, Joachim Schiele, Joe Hermaszewski, Joel Moberg, Johannes 'fish' Ziemke, Jörg Thalheim, Jude Taylor, kballou, Keshav Kini, Kjetil Orbekk, Langston Barrett, Linus Heckemann, Ludovic Courtès, Manav Rathi, Marc Scholten, Markus Hauck, Matt Audesse, Matthew Bauer, Matthias Beyer, Matthieu Coudron, N1X, Nathan Zadoks, Neil Mayhew, Nicolas B. Pierron, Niklas Hambüchen, Nikolay Amiantov, Ole Jørgen Brønner, Orivej Desh, Peter Simons, Peter Stuart, Pyry Jahkola, regnat, Renzo Carbonara, Rhys, Robert Vollmert, Scott Olson, Scott R. Parish, Sergei Trofimovich, Shea Levy, Sheena Artrip, Spencer Baugh, Stefan Junker, Susan Potter, Thomas Tuegel, Timothy Allen, Tristan Hume, Tuomas Tynkkynen, tv, Tyson Whitehead, Vladimír Čunát, Will Dietz, wmertens, Wout Mertens, zimbatm and Zoran Plesivčak.

# Release 1.11.10 (2017-06-12)

This release fixes a security bug in Nix’s “build user” build isolation mechanism. Previously, Nix builders had the ability to create setuid binaries owned by a `nixbld` user. Such a binary could then be used by an attacker to assume a `nixbld` identity and interfere with subsequent builds running under the same UID.

To prevent this issue, Nix now disallows builders to create setuid and setgid binaries. On Linux, this is done using a seccomp BPF filter. Note that this imposes a small performance penalty (e.g. 1% when building GNU Hello). Using seccomp, we now also prevent the creation of extended attributes and POSIX ACLs since these cannot be represented in the NAR format and (in the case of POSIX ACLs) allow bypassing regular Nix store permissions. On macOS, the restriction is implemented using the existing sandbox mechanism, which now uses a minimal “allow all except the creation of setuid/setgid binaries” profile when regular sandboxing is disabled. On other platforms, the “build user” mechanism is now disabled.

Thanks go to Linus Heckemann for discovering and reporting this bug.

# Release 1.11 (2016-01-19)

This is primarily a bug fix release. It also has a number of new features:

- `nix-prefetch-url` can now download URLs specified in a Nix expression. For example,

```
$ nix-prefetch-url -A hello.src
```

will prefetch the file specified by the `fetchurl` call in the attribute `hello.src` from the Nix expression in the current directory, and print the cryptographic hash of the resulting file on stdout. This differs from `nix-build -A hello.src` in that it doesn't verify the hash, and is thus useful when you're updating a Nix expression.

You can also prefetch the result of functions that unpack a tarball, such as `fetchFromGitHub`. For example:

```
$ nix-prefetch-url --unpack https://github.com/NixOS/patchelf/archive/0.8.tar.gz
```

or from a Nix expression:

```
$ nix-prefetch-url -A nix-repl.src
```

- The builtin function `<nix/fetchurl.nix>` now supports downloading and unpacking NARs. This removes the need to have multiple downloads in the Nixpkgs stdenv bootstrap process (like a separate busybox binary for Linux, or curl/mkdir/sh/bzip2 for Darwin). Now all those files can be combined into a single NAR, optionally compressed using `xz`.
- Nix now supports SHA-512 hashes for verifying fixed-output derivations, and in `builtins.hashString`.
- The new flag `--option build-repeat N` will cause every build to be executed  $N+1$  times. If the build output differs between any round, the build is rejected, and the output paths are not registered as valid. This is primarily useful to verify build determinism. (We already had a `--check` option to repeat a previously succeeded build. However, with `--check`, non-deterministic builds are registered in the DB. Preventing that is useful for Hydra to ensure that non-deterministic builds don't end up getting published to the binary cache.)

- The options `--check` and `--option build-repeat N`, if they detect a difference between two runs of the same derivation and `-K` is given, will make the output of the other run available under `store-path-check`. This makes it easier to investigate the non-determinism using tools like `diffoscope`, e.g.,

```
$ nix-build pkgs/stdenv/linux -A stage1.pkgs.zlib --check -K
error: derivation '/nix/store/l54i8wlw2265...-zlib-1.2.8.drv' may not
be deterministic: output '/nix/store/11a27shh6n2i...-zlib-1.2.8'
differs from '/nix/store/11a27shh6n2i...-zlib-1.2.8-check'
```

```
$ diffoscope /nix/store/11a27shh6n2i...-zlib-1.2.8 /nix/store/11a27shh6n2i...-zlib-1.2.8-check
```

```
...
├── lib/libz.a
│ ├── metadata
│ │ @@ -1,15 +1,15 @@
│ │ -rw-r--r-- 30001/30000 3096 Jan 12 15:20 2016 adler32.o
...
| | +rw-r--r-- 30001/30000 3096 Jan 12 15:28 2016 adler32.o
...

```

- Improved FreeBSD support.
  - `nix-env -qa --xml --meta` now prints license information.
  - The maximum number of parallel TCP connections that the binary cache substituter will use has been decreased from 150 to 25. This should prevent upsetting some broken NAT routers, and also improves performance.
  - All "chroot"-containing strings got renamed to "sandbox". In particular, some Nix options got renamed, but the old names are still accepted as lower-priority aliases.

This release has contributions from Anders Claesson, Anthony Cowley, Bjørn Forsman, Brian McKenna, Danny Wilson, davidak, Eelco Dolstra, Fabian Schmitthenner, FrankHB, Ilya Novoselov, janus, Jim Garrison, John Ericson, Jude Taylor, Ludovic Courtès, Manuel Jacob, Mathnerd314, Pascal Wittmann, Peter Simons, Philip Potter, Preston Bennes, Rommel M. Martinez, Sander van der Burg, Shea Levy, Tim Cuthbertson, Tuomas Tynkkynen, Utku Demir and Vladimír Čunát.

# Release 1.10 (2015-09-03)

This is primarily a bug fix release. It also has a number of new features:

- A number of builtin functions have been added to reduce Nixpkgs/NixOS evaluation time and memory consumption: `all`, `any`, `concatStringsSep`, `foldl'`, `genList`, `replaceStrings`, `sort`.
- The garbage collector is more robust when the disk is full.
- Nix supports a new API for building derivations that doesn't require a `.drv` file to be present on disk; it only requires an in-memory representation of the derivation. This is used by the Hydra continuous build system to make remote builds more efficient.
- The function `<nix/fetchurl.nix>` now uses a *builtin* builder (i.e. it doesn't require starting an external process; the download is performed by Nix itself). This ensures that derivation paths don't change when Nix is upgraded, and obviates the need for ugly hacks to support chroot execution.
- `--version -v` now prints some configuration information, in particular what compile-time optional features are enabled, and the paths of various directories.
- Build users have their supplementary groups set correctly.

This release has contributions from Eelco Dolstra, Guillaume Maudoux, Iwan Aucamp, Jaka Hudoklin, Kirill Elagin, Ludovic Courtès, Manolis Ragkousis, Nicolas B. Pierron and Shea Levy.

# Release 1.9 (2015-06-12)

In addition to the usual bug fixes, this release has the following new features:

- Signed binary cache support. You can enable signature checking by adding the following to `nix.conf`:

```
signed-binary-caches = *
binary-cache-public-keys = cache.nixos.org-
1:6NCHdD59X431o0gWypbMrAURkbJ16ZPMQFGspcDShjY=
```

This will prevent Nix from downloading any binary from the cache that is not signed by one of the keys listed in `binary-cache-public-keys`.

Signature checking is only supported if you built Nix with the `libsodium` package.

Note that while Nix has had experimental support for signed binary caches since version 1.7, this release changes the signature format in a backwards-incompatible way.

- Automatic downloading of Nix expression tarballs. In various places, you can now specify the URL of a tarball containing Nix expressions (such as `Nixpkgs`), which will be downloaded and unpacked automatically. For example:

- In `nix-env`:

```
$ nix-env -f https://github.com/NixOS/nixpkgs-channels/archive/nixos-
14.12.tar.gz -iA firefox
```

This installs Firefox from the latest tested and built revision of the NixOS 14.12 channel.

- In `nix-build` and `nix-shell`:

```
$ nix-build https://github.com/NixOS/nixpkgs/archive/master.tar.gz -A
hello
```

This builds GNU Hello from the latest revision of the `Nixpkgs` master branch.

- In the Nix search path (as specified via `NIX_PATH` or `-I`). For example, to start a shell containing the Pan package from a specific version of `Nixpkgs`:

```
$ nix-shell -p pan -I nixpkgs=https://github.com/NixOS/nixpkgs-channels/archive/8a3eea054838b55aca962c3fbde9c83c102b8bf2.tar.gz
```

- In `nixos-rebuild` (on NixOS):

```
$ nixos-rebuild test -I nixpkgs=https://github.com/NixOS/nixpkgs-channels/archive/nixos-unstable.tar.gz
```

- In Nix expressions, via the new builtin function `fetchTarball`:

```
with import (fetchTarball https://github.com/NixOS/nixpkgs-channels/archive/nixos-14.12.tar.gz) {}; ...
```

(This is not allowed in restricted mode.)

- `nix-shell` improvements:

- `nix-shell` now has a flag `--run` to execute a command in the `nix-shell` environment, e.g. `nix-shell --run make`. This is like the existing `--command` flag, except that it uses a non-interactive shell (ensuring that hitting Ctrl-C won't drop you into the child shell).
- `nix-shell` can now be used as a `#!`-interpreter. This allows you to write scripts that dynamically fetch their own dependencies. For example, here is a Haskell script that, when invoked, first downloads GHC and the Haskell packages on which it depends:

```
#!/usr/bin/env nix-shell
#! nix-shell -i runghc -p haskellPackages.ghc haskellPackages.HTTP

import Network.HTTP

main = do
 resp <- Network.HTTP.simpleHTTP (getRequest "http://nixos.org/")
 body <- getResponseBody resp
 print (take 100 body)
```

Of course, the dependencies are cached in the Nix store, so the second invocation of this script will be much faster.

- Chroot improvements:

- Chroot builds are now supported on Mac OS X (using its sandbox mechanism).
  - If chroots are enabled, they are now used for all derivations, including fixed-output derivations (such as `fetchurl`). The latter do have network access, but can no longer access the host filesystem. If you need the old behaviour, you can set the option `build-use-chroot` to `relaxed`.
  - On Linux, if chroots are enabled, builds are performed in a private PID namespace once again. (This functionality was lost in Nix 1.8.)
  - Store paths listed in `build-chroot-dirs` are now automatically expanded to their closure. For instance, if you want `/nix/store/...-bash/bin/sh` mounted in your chroot as `/bin/sh`, you only need to say `build-chroot-dirs = /bin/sh= /nix/store/...-bash/bin/sh`; it is no longer necessary to specify the dependencies of Bash.
- The new derivation attribute `passAsFile` allows you to specify that the contents of derivation attributes should be passed via files rather than environment variables. This is useful if you need to pass very long strings that exceed the size limit of the environment. The `Nixpkgs` function `writeTextFile` uses this.
  - You can now use `~` in Nix file names to refer to your home directory, e.g. `import ~/.nixpkgs/config.nix`.
  - Nix has a new option `restrict-eval` that allows limiting what paths the Nix evaluator has access to. By passing `--option restrict-eval true` to Nix, the evaluator will throw an exception if an attempt is made to access any file outside of the Nix search path. This is primarily intended for Hydra to ensure that a Hydra jobset only refers to its declared inputs (and is therefore reproducible).
  - `nix-env` now only creates a new “generation” symlink in `/nix/var/nix/profiles` if something actually changed.
  - The environment variable `NIX_PAGER` can now be set to override `PAGER`. You can set it to `cat` to disable paging for Nix commands only.
  - Failing `<...>` lookups now show position information.
  - Improved Boehm GC use: we disabled scanning for interior pointers, which should reduce the “`Repeated allocation of very large block`” warnings and associated retention of memory.

This release has contributions from aszlig, Benjamin Staffin, Charles Strahan, Christian Theune, Daniel Hahler, Danylo Hlynksyi Daniel Peebles, Dan Peebles, Domen Kožar, Eelco

Dolstra, Harald van Dijk, Hoang Xuan Phu, Jaka Hudoklin, Jeff Ramnani, j-keck, Linquize, Luca Bruno, Michael Merickel, Oliver Dunkl, Rob Vermaas, Rok Garbas, Shea Levy, Tobias Geerinckx-Rice and William A. Kennington III.

# Release 1.8 (2014-12-14)

- Breaking change: to address a race condition, the remote build hook mechanism now uses `nix-store --serve` on the remote machine. This requires build slaves to be updated to Nix 1.8.
- Nix now uses HTTPS instead of HTTP to access the default binary cache, `cache.nixos.org`.
- `nix-env` selectors are now regular expressions. For instance, you can do

```
$ nix-env -qa '.*zip.*'
```

to query all packages with a name containing `zip`.

- `nix-store --read-log` can now fetch remote build logs. If a build log is not available locally, then ‘`nix-store -l`’ will now try to download it from the servers listed in the ‘log-servers’ option in `nix.conf`. For instance, if you have the configuration option

```
log-servers = http://hydra.nixos.org/log
```

then it will try to get logs from `http://hydra.nixos.org/log/base name of the store path`. This allows you to do things like:

```
$ nix-store -l $(which xterm)
```

and get a log even if `xterm` wasn’t built locally.

- New builtin functions: `attrValues`, `deepSeq`, `fromJSON`, `readDir`, `seq`.
- `nix-instantiate --eval` now has a `--json` flag to print the resulting value in JSON format.
- `nix-copy-closure` now uses `nix-store --serve` on the remote side to send or receive closures. This fixes a race condition between `nix-copy-closure` and the garbage collector.
- Derivations can specify the new special attribute `allowedRequisites`, which has a similar meaning to `allowedReferences`. But instead of only enforcing to explicitly specify the immediate references, it requires the derivation to specify all the dependencies recursively (hence the name, requisites) that are used by the resulting

output.

- On Mac OS X, Nix now handles case collisions when importing closures from case-sensitive file systems. This is mostly useful for running NixOps on Mac OS X.
- The Nix daemon has new configuration options `allowed-users` (specifying the users and groups that are allowed to connect to the daemon) and `trusted-users` (specifying the users and groups that can perform privileged operations like specifying untrusted binary caches).
- The configuration option `build-cores` now defaults to the number of available CPU cores.
- Build users are now used by default when Nix is invoked as root. This prevents builds from accidentally running as root.
- Nix now includes systemd units and Upstart jobs.
- Speed improvements to `nix-store --optimise`.
- Language change: the `==` operator now ignores string contexts (the “dependencies” of a string).
- Nix now filters out Nix-specific ANSI escape sequences on standard error. They are supposed to be invisible, but some terminals show them anyway.
- Various commands now automatically pipe their output into the pager as specified by the `PAGER` environment variable.
- Several improvements to reduce memory consumption in the evaluator.

This release has contributions from Adam Szkoda, Aristid Breitkreuz, Bob van der Linden, Charles Strahan, darealshinji, Eelco Dolstra, Gergely Risko, Joel Taylor, Ludovic Courtès, Marko Durkovic, Mikey Ariel, Paul Colomiets, Ricardo M. Correia, Ricky Elrod, Robert Helgesson, Rob Vermaas, Russell O'Connor, Shea Levy, Shell Turner, Sönke Hahn, Steve Purcell, Vladimír Čunát and Wout Mertens.

# Release 1.7 (2014-04-11)

In addition to the usual bug fixes, this release has the following new features:

- Antiquotation is now allowed inside of quoted attribute names (e.g. `set."${foo}"`). In the case where the attribute name is just a single antiquotation, the quotes can be dropped (e.g. the above example can be written `set.${foo}`). If an attribute name inside of a set declaration evaluates to `null` (e.g. `{ ${null} = false; }`), then that attribute is not added to the set.
- Experimental support for cryptographically signed binary caches. See [the commit for details](#).
- An experimental new substituter, `download-via-ssh`, that fetches binaries from remote machines via SSH. Specifying the flags `--option use-ssh-substituter true` `--option ssh-substituter-hosts user@hostname` will cause Nix to download binaries from the specified machine, if it has them.
- `nix-store -r` and `nix-build` have a new flag, `--check`, that builds a previously built derivation again, and prints an error message if the output is not exactly the same. This helps to verify whether a derivation is truly deterministic. For example:

```
$ nix-build '<nixpkgs>' -A patchelf
...
$ nix-build '<nixpkgs>' -A patchelf --check
...
error: derivation `/nix/store/1ipvxs...-patchelf-0.6' may not be
deterministic:
hash mismatch in output `/nix/store/4pc1dm...-patchelf-0.6.drv'
```

- The `nix-instantiate` flags `--eval-only` and `--parse-only` have been renamed to `--eval` and `--parse`, respectively.
- `nix-instantiate`, `nix-build` and `nix-shell` now have a flag `--expr` (or `-E`) that allows you to specify the expression to be evaluated as a command line argument. For instance, `nix-instantiate --eval -E '1 + 2'` will print `3`.
- `nix-shell` improvements:
  - It has a new flag, `--packages` (or `-p`), that sets up a build environment containing the specified packages from Nixpkgs. For example, the command

```
$ nix-shell -p sqlite xorg.libX11 hello
```

will start a shell in which the given packages are present.

- It now uses `shell.nix` as the default expression, falling back to `default.nix` if the former doesn’t exist. This makes it convenient to have a `shell.nix` in your project to set up a nice development environment.
  - It evaluates the derivation attribute `shellHook`, if set. Since `stdenv` does not normally execute this hook, it allows you to do `nix-shell`-specific setup.
  - It preserves the user’s timezone setting.
- In chroots, Nix now sets up a `/dev` containing only a minimal set of devices (such as `/dev/null`). Note that it only does this if you *don’t* have `/dev` listed in your `build-chroot-dirs` setting; otherwise, it will bind-mount the `/dev` from outside the chroot. Similarly, if you don’t have `/dev/pts` listed in `build-chroot-dirs`, Nix will mount a private `devpts` filesystem on the chroot’s `/dev/pts`.
  - New built-in function: `builtins.toJSON`, which returns a JSON representation of a value.
  - `nix-env -q` has a new flag `--json` to print a JSON representation of the installed or available packages.
  - `nix-env` now supports meta attributes with more complex values, such as attribute sets.
  - The `-A` flag now allows attribute names with dots in them, e.g.

```
$ nix-instantiate --eval '<nixos>' -A
'config.systemd.units."nscd.service".text'
```

- The `--max-freed` option to `nix-store --gc` now accepts a unit specifier. For example, `nix-store --gc --max-freed 1G` will free up to 1 gigabyte of disk space.
- `nix-collect-garbage` has a new flag `--delete-older-than N d`, which deletes all user environment generations older than  $N$  days. Likewise, `nix-env --delete-generations` accepts a  $N$  age limit.
- Nix now heuristically detects whether a build failure was due to a disk-full condition. In that case, the build is not flagged as “permanently failed”. This is mostly useful for

Hydra, which needs to distinguish between permanent and transient build failures.

- There is a new symbol `__curPos` that expands to an attribute set containing its file name and line and column numbers, e.g. `{ file = "foo.nix"; line = 10; column = 5; }`. There also is a new builtin function, `unsafeGetAttrPos`, that returns the position of an attribute. This is used by Nixpkgs to provide location information in error messages, e.g.

```
$ nix-build '<nixpkgs>' -A libreoffice --argstr system x86_64-darwin
error: the package 'libreoffice-4.0.5.2' in '.../applications/office
/libreoffice/default.nix:263'
 is not supported on 'x86_64-darwin'
```

- The garbage collector is now more concurrent with other Nix processes because it releases certain locks earlier.
- The binary tarball installer has been improved. You can now install Nix by running:

```
$ bash <(curl -L https://nixos.org/nix/install)
```

- More evaluation errors include position information. For instance, selecting a missing attribute will print something like

```
error: attribute `nixUnstabl' missing, at /etc/nixos/configurations
/misc/eelco/mandark.nix:216:15
```

- The command `nix-setuid-helper` is gone.
- Nix no longer uses Automake, but instead has a non-recursive, GNU Make-based build system.
- All installed libraries now have the prefix `libnix`. In particular, this gets rid of `libutil`, which could clash with libraries with the same name from other packages.
- Nix now requires a compiler that supports C++11.

This release has contributions from Danny Wilson, Domen Kožar, Eelco Dolstra, Ian-Woo Kim, Ludovic Courtès, Maxim Ivanov, Petr Rockai, Ricardo M. Correia and Shea Levy.

# Release 1.6.1 (2013-10-28)

This is primarily a bug fix release. Changes of interest are:

- Nix 1.6 accidentally changed the semantics of antiquoted paths in strings, such as `"${{/foo}}/bar"`. This release reverts to the Nix 1.5.3 behaviour.
- Previously, Nix optimised expressions such as `"${{expr}}"` to `expr`. Thus it neither checked whether `expr` could be coerced to a string, nor applied such coercions. This meant that `"${{123}}"` evaluated to `123`, and `"${{./foo}}"` evaluated to `./foo` (even though `"${{./foo}}"` evaluates to `"/nix/store/hash-foo "`). Nix now checks the type of antiquoted expressions and applies coercions.
- Nix now shows the exact position of undefined variables. In particular, undefined variable errors in a `with` previously didn't show *any* position information, so this makes it a lot easier to fix such errors.
- Undefined variables are now treated consistently. Previously, the `tryEval` function would catch undefined variables inside a `with` but not outside. Now `tryEval` never catches undefined variables.
- Bash completion in `nix-shell` now works correctly.
- Stack traces are less verbose: they no longer show calls to builtin functions and only show a single line for each derivation on the call stack.
- New built-in function: `builtins.typeOf`, which returns the type of its argument as a string.

# Release 1.6 (2013-09-10)

In addition to the usual bug fixes, this release has several new features:

- The command `nix-build --run-env` has been renamed to `nix-shell`.
- `nix-shell` now sources `$stdenv/setup` *inside* the interactive shell, rather than in a parent shell. This ensures that shell functions defined by `stdenv` can be used in the interactive shell.
- `nix-shell` has a new flag `--pure` to clear the environment, so you get an environment that more closely corresponds to the “real” Nix build.
- `nix-shell` now sets the shell prompt (`PS1`) to ensure that Nix shells are distinguishable from your regular shells.
- `nix-env` no longer requires a `*` argument to match all packages, so `nix-env -qa` is equivalent to `nix-env -qa '*'`.
- `nix-env -i` has a new flag `--remove-all` (`-r`) to remove all previous packages from the profile. This makes it easier to do declarative package management similar to NixOS’s `environment.systemPackages`. For instance, if you have a specification `my-packages.nix` like this:

```
with import <nixpkgs> {};
[thunderbird
 geeqie
 ...
]
```

then after any change to this file, you can run:

```
$ nix-env -f my-packages.nix -ir
```

to update your profile to match the specification.

- The ‘`with`’ language construct is now more lazy. It only evaluates its argument if a variable might actually refer to an attribute in the argument. For instance, this now works:

```
let
 pkgs = with pkgs; { foo = "old"; bar = foo; } // overrides;
 overrides = { foo = "new"; };
in pkgs.bar
```

This evaluates to `"new"`, while previously it gave an “infinite recursion” error.

- Nix now has proper integer arithmetic operators. For instance, you can write `x + y` instead of `builtins.add x y`, or `x < y` instead of `builtins.lessThan x y`. The comparison operators also work on strings.
- On 64-bit systems, Nix integers are now 64 bits rather than 32 bits.
- When using the Nix daemon, the `nix-daemon` worker process now runs on the same CPU as the client, on systems that support setting CPU affinity. This gives a significant speedup on some systems.
- If a stack overflow occurs in the Nix evaluator, you now get a proper error message (rather than “Segmentation fault”) on some systems.
- In addition to directories, you can now bind-mount regular files in chroots through the (now misnamed) option `build-chroot-dirs`.

This release has contributions from Domen Kožar, Eelco Dolstra, Florian Friesdorf, Gergely Risko, Ivan Kozik, Ludovic Courtès and Shea Levy.

# Release 1.5.2 (2013-05-13)

This is primarily a bug fix release. It has contributions from Eelco Dolstra, Lluís Batlle i Rossell and Shea Levy.

# Release 1.5 (2013-02-27)

This is a brown paper bag release to fix a regression introduced by the hard link security fix in 1.4.

# Release 1.4 (2013-02-26)

This release fixes a security bug in multi-user operation. It was possible for derivations to cause the mode of files outside of the Nix store to be changed to 444 (read-only but world-readable) by creating hard links to those files ([details](#)).

There are also the following improvements:

- New built-in function: `builtins.hashString`.
- Build logs are now stored in `/nix/var/log/nix/drvs/XX/`, where XX is the first two characters of the derivation. This is useful on machines that keep a lot of build logs (such as Hydra servers).
- The function `corepkgs/fetchurl` can now make the downloaded file executable. This will allow getting rid of all bootstrap binaries in the Nixpkgs source tree.
- Language change: The expression `"${./path} ..."` now evaluates to a string instead of a path.

# Release 1.3 (2013-01-04)

This is primarily a bug fix release. When this version is first run on Linux, it removes any immutable bits from the Nix store and increases the schema version of the Nix store. (The previous release removed support for setting the immutable bit; this release clears any remaining immutable bits to make certain operations more efficient.)

This release has contributions from Eelco Dolstra and Stuart Pernsteiner.

# Release 1.2 (2012-12-06)

This release has the following improvements and changes:

- Nix has a new binary substituter mechanism: the *binary cache*. A binary cache contains pre-built binaries of Nix packages. Whenever Nix wants to build a missing Nix store path, it will check a set of binary caches to see if any of them has a pre-built binary of that path. The configuration setting `binary-caches` contains a list of URLs of binary caches. For instance, doing

```
$ nix-env -i thunderbird --option binary-caches http://cache.nixos.org
```

will install Thunderbird and its dependencies, using the available pre-built binaries in <http://cache.nixos.org>. The main advantage over the old “manifest”-based method of getting pre-built binaries is that you don’t have to worry about your manifest being in sync with the Nix expressions you’re installing from; i.e., you don’t need to run `nix-pull` to update your manifest. It’s also more scalable because you don’t need to redownload a giant manifest file every time.

A Nix channel can provide a binary cache URL that will be used automatically if you subscribe to that channel. If you use the Nixpkgs or NixOS channels (<http://nixos.org/channels>) you automatically get the cache <http://cache.nixos.org>.

Binary caches are created using `nix-push`. For details on the operation and format of binary caches, see the `nix-push` manpage. More details are provided in [this nix-dev posting](#).

- Multiple output support should now be usable. A derivation can declare that it wants to produce multiple store paths by saying something like

```
outputs = ["lib" "headers" "doc"];
```

This will cause Nix to pass the intended store path of each output to the builder through the environment variables `lib`, `headers` and `doc`. Other packages can refer to a specific output by referring to `pkg.output`, e.g.

```
buildInputs = [pkg.lib pkg.headers];
```

If you install a package with multiple outputs using `nix-env`, each output path will be symlinked into the user environment.

- Dashes are now valid as part of identifiers and attribute names.
- The new operation `nix-store --repair-path` allows corrupted or missing store paths to be repaired by redownloading them. `nix-store --verify --check-contents --repair` will scan and repair all paths in the Nix store. Similarly, `nix-env`, `nix-build`, `nix-instantiate` and `nix-store --realise` have a `--repair` flag to detect and fix bad paths by rebuilding or redownloading them.
- Nix no longer sets the immutable bit on files in the Nix store. Instead, the recommended way to guard the Nix store against accidental modification on Linux is to make it a read-only bind mount, like this:

```
$ mount --bind /nix/store /nix/store
$ mount -o remount,ro,bind /nix/store
```

Nix will automatically make `/nix/store` writable as needed (using a private mount namespace) to allow modifications.

- Store optimisation (replacing identical files in the store with hard links) can now be done automatically every time a path is added to the store. This is enabled by setting the configuration option `auto-optimise-store` to `true` (disabled by default).
- Nix now supports `xz` compression for NARs in addition to `bzip2`. It compresses about 30% better on typical archives and decompresses about twice as fast.
- Basic Nix expression evaluation profiling: setting the environment variable `NIX_COUNT_CALLS` to `1` will cause Nix to print how many times each primop or function was executed.
- New primops: `concatLists`, `elem`, `elemAt` and `filter`.
- The command `nix-copy-closure` has a new flag `--use-substitutes` (`-s`) to download missing paths on the target machine using the substitute mechanism.
- The command `nix-worker` has been renamed to `nix-daemon`. Support for running the Nix worker in “slave” mode has been removed.
- The `--help` flag of every Nix command now invokes `man`.
- Chroot builds are now supported on systemd machines.

This release has contributions from Eelco Dolstra, Florian Friesdorf, Mats Erik Andersson and Shea Levy.

# Release 1.1 (2012-07-18)

This release has the following improvements:

- On Linux, when doing a chroot build, Nix now uses various namespace features provided by the Linux kernel to improve build isolation. Namely:
  - The private network namespace ensures that builders cannot talk to the outside world (or vice versa): each build only sees a private loopback interface. This also means that two concurrent builds can listen on the same port (e.g. as part of a test) without conflicting with each other.
  - The PID namespace causes each build to start as PID 1. Processes outside of the chroot are not visible to those on the inside. On the other hand, processes inside the chroot *are* visible from the outside (though with different PIDs).
  - The IPC namespace prevents the builder from communicating with outside processes using SysV IPC mechanisms (shared memory, message queues, semaphores). It also ensures that all IPC objects are destroyed when the builder exits.
  - The UTS namespace ensures that builders see a hostname of `localhost` rather than the actual hostname.
  - The private mount namespace was already used by Nix to ensure that the bind-mounts used to set up the chroot are cleaned up automatically.
- Build logs are now compressed using `bzip2`. The command `nix-store -l` decompresses them on the fly. This can be disabled by setting the option `build-compress-log` to `false`.
- The creation of build logs in `/nix/var/log/nix/drvs` can be disabled by setting the new option `build-keep-log` to `false`. This is useful, for instance, for Hydra build machines.
- Nix now reserves some space in `/nix/var/nix/db/reserved` to ensure that the garbage collector can run successfully if the disk is full. This is necessary because SQLite transactions fail if the disk is full.
- Added a basic `fetchurl` function. This is not intended to replace the `fetchurl` in Nixpkgs, but is useful for bootstrapping; e.g., it will allow us to get rid of the bootstrap binaries in the Nixpkgs source tree and download them instead. You can use it by doing `import <nix/fetchurl.nix> { url = url; sha256 = "hash"; }`. (Shea Levy)

- Improved RPM spec file. (Michel Alexandre Salim)
- Support for on-demand socket-based activation in the Nix daemon with `systemd`.
- Added a manpage for `nix.conf5`.
- When using the Nix daemon, the `-s` flag in `nix-env -qa` is now much faster.

# Release 1.0 (2012-05-11)

There have been numerous improvements and bug fixes since the previous release. Here are the most significant:

- Nix can now optionally use the Boehm garbage collector. This significantly reduces the Nix evaluator's memory footprint, especially when evaluating large NixOS system configurations. It can be enabled using the `--enable-gc` configure option.
- Nix now uses SQLite for its database. This is faster and more flexible than the old *ad hoc* format. SQLite is also used to cache the manifests in `/nix/var/nix/manifests`, resulting in a significant speedup.
- Nix now has a search path for expressions. The search path is set using the environment variable `NIX_PATH` and the `-I` command line option. In Nix expressions, paths between angle brackets are used to specify files that must be looked up in the search path. For instance, the expression `<nixpkgs/default.nix>` looks for a file `nixpkgs/default.nix` relative to every element in the search path.
- The new command `nix-build --run-env` builds all dependencies of a derivation, then starts a shell in an environment containing all variables from the derivation. This is useful for reproducing the environment of a derivation for development.
- The new command `nix-store --verify-path` verifies that the contents of a store path have not changed.
- The new command `nix-store --print-env` prints out the environment of a derivation in a format that can be evaluated by a shell.
- Attribute names can now be arbitrary strings. For instance, you can write `{ "foo-1.2" = ...; "bla bla" = ...; }."bla bla".`
- Attribute selection can now provide a default value using the `or` operator. For instance, the expression `x.y.z or e` evaluates to the attribute `x.y.z` if it exists, and `e` otherwise.
- The right-hand side of the `?` operator can now be an attribute path, e.g., `attrs ? a.b.c .`
- On Linux, Nix will now make files in the Nix store immutable on filesystems that support it. This prevents accidental modification of files in the store by the root user.
- Nix has preliminary support for derivations with multiple outputs. This is useful

because it allows parts of a package to be deployed and garbage-collected separately. For instance, development parts of a package such as header files or static libraries would typically not be part of the closure of an application, resulting in reduced disk usage and installation time.

- The Nix store garbage collector is faster and holds the global lock for a shorter amount of time.
- The option `--timeout` (corresponding to the configuration setting `build-timeout`) allows you to set an absolute timeout on builds — if a build runs for more than the given number of seconds, it is terminated. This is useful for recovering automatically from builds that are stuck in an infinite loop but keep producing output, and for which `--max-silent-time` is ineffective.
- Nix development has moved to GitHub (<https://github.com/NixOS/nix>).

# Release 0.16 (2010-08-17)

This release has the following improvements:

- The Nix expression evaluator is now much faster in most cases: typically, [3 to 8 times compared to the old implementation](#). It also uses less memory. It no longer depends on the ATerm library.
- Support for configurable parallelism inside builders. Build scripts have always had the ability to perform multiple build actions in parallel (for instance, by running `make -j 2`), but this was not desirable because the number of actions to be performed in parallel was not configurable. Nix now has an option `--cores N` as well as a configuration setting `build-cores = N` that causes the environment variable `NIX_BUILD_CORES` to be set to  $N$  when the builder is invoked. The builder can use this at its discretion to perform a parallel build, e.g., by calling `make -j N`. In Nixpkgs, this can be enabled on a per-package basis by setting the derivation attribute `enableParallelBuilding` to `true`.
- `nix-store -q` now supports XML output through the `--xml` flag.
- Several bug fixes.

# Release 0.15 (2010-03-17)

This is a bug-fix release. Among other things, it fixes building on Mac OS X (Snow Leopard), and improves the contents of `/etc/passwd` and `/etc/group` in `chroot` builds.

# Release 0.14 (2010-02-04)

This release has the following improvements:

- The garbage collector now starts deleting garbage much faster than before. It no longer determines liveness of all paths in the store, but does so on demand.
- Added a new operation, `nix-store --query --roots`, that shows the garbage collector roots that directly or indirectly point to the given store paths.
- Removed support for converting Berkeley DB-based Nix databases to the new schema.
- Removed the `--use-ctime` and `--max-ctime` garbage collector options. They were not very useful in practice.
- On Windows, Nix now requires Cygwin 1.7.x.
- A few bug fixes.

# Release 0.13 (2009-11-05)

This is primarily a bug fix release. It has some new features:

- Syntactic sugar for writing nested attribute sets. Instead of

```
{
 foo = {
 bar = 123;
 xyzzy = true;
 };
 a = { b = { c = "d"; }; };
}
```

you can write

```
{
 foo.bar = 123;
 foo.xyzzy = true;
 a.b.c = "d";
}
```

This is useful, for instance, in NixOS configuration files.

- Support for Nix channels generated by Hydra, the Nix-based continuous build system. (Hydra generates NAR archives on the fly, so the size and hash of these archives isn't known in advance.)
- Support `i686-linux` builds directly on `x86_64-linux` Nix installations. This is implemented using the `personality()` syscall, which causes `uname` to return `i686` in child processes.
- Various improvements to the `chroot` support. Building in a `chroot` works quite well now.
- Nix no longer blocks if it tries to build a path and another process is already building the same path. Instead it tries to build another buildable path first. This improves parallelism.
- Support for large (> 4 GiB) files in NAR archives.
- Various (performance) improvements to the remote build mechanism.

- New primops: `builtins.addErrorContext` (to add a string to stack traces — useful for debugging), `builtins.isBool`, `builtins.isString`, `builtins.toInt`, `builtins.intersectAttrs`.
- OpenSolaris support (Sander van der Burg).
- Stack traces are no longer displayed unless the `--show-trace` option is used.
- The scoping rules for `inherit (e) ...` in recursive attribute sets have changed. The expression `e` can now refer to the attributes defined in the containing set.

# Release 0.12 (2008-11-20)

- Nix no longer uses Berkeley DB to store Nix store metadata. The principal advantages of the new storage scheme are: it works properly over decent implementations of NFS (allowing Nix stores to be shared between multiple machines); no recovery is needed when a Nix process crashes; no write access is needed for read-only operations; no more running out of Berkeley DB locks on certain operations.

You still need to compile Nix with Berkeley DB support if you want Nix to automatically convert your old Nix store to the new schema. If you don't need this, you can build Nix with the `configure` option `--disable-old-db-compat`.

After the automatic conversion to the new schema, you can delete the old Berkeley DB files:

```
$ cd /nix/var/nix/db
$ rm __db* log.* deriviers references referrers reserved validpaths
DB_CONFIG
```

The new metadata is stored in the directories `/nix/var/nix/db/info` and `/nix/var/nix/db/referrer`. Though the metadata is stored in human-readable plain-text files, they are not intended to be human-editable, as Nix is rather strict about the format.

The new storage schema may or may not require less disk space than the Berkeley DB environment, mostly depending on the cluster size of your file system. With 1 KiB clusters (which seems to be the `ext3` default nowadays) it usually takes up much less space.

- There is a new substituter that copies paths directly from other (remote) Nix stores mounted somewhere in the filesystem. For instance, you can speed up an installation by mounting some remote Nix store that already has the packages in question via NFS or `sshfs`. The environment variable `NIX_OTHER_STORES` specifies the locations of the remote Nix directories, e.g. `/mnt/remote-fs/nix`.
- New `nix-store` operations `--dump-db` and `--load-db` to dump and reload the Nix database.
- The garbage collector has a number of new options to allow only some of the garbage to be deleted. The option `--max-freed N` tells the collector to stop after at least  $N$  bytes have been deleted. The option `--max-links N` tells it to stop after the link count on `/nix/store` has dropped below  $N$ . This is useful for very large Nix stores on

filesystems with a 32000 subdirectories limit (like `ext3`). The option `--use-atime` causes store paths to be deleted in order of ascending last access time. This allows non-recently used stuff to be deleted. The option `--max-atime time` specifies an upper limit to the last accessed time of paths that may be deleted. For instance,

```
$ nix-store --gc -v --max-atime $(date +%s -d "2 months ago")
```

deletes everything that hasn't been accessed in two months.

- `nix-env` now uses optimistic profile locking when performing an operation like installing or upgrading, instead of setting an exclusive lock on the profile. This allows multiple `nix-env -i` / `-u` / `-e` operations on the same profile in parallel. If a `nix-env` operation sees at the end that the profile was changed in the meantime by another process, it will just restart. This is generally cheap because the build results are still in the Nix store.
- The option `--dry-run` is now supported by `nix-store -r` and `nix-build`.
- The information previously shown by `--dry-run` (i.e., which derivations will be built and which paths will be substituted) is now always shown by `nix-env`, `nix-store -r` and `nix-build`. The total download size of substitutable paths is now also shown. For instance, a build will show something like

the following derivations will be built:

```
/nix/store/129sbxnk5n466zg6r1qmq1xjv9zymyy7-activate-configuration.shdrv
/nix/store/7mzy971rdm8l566ch8hgxaf89x7lr7ik-upstart-jobs.drv
...
```

the following paths will be downloaded/copied (30.02 MiB):

```
/nix/store/4m8pvgv2dcjgpff5b4cj5l6wyshjhaj-samba-3.2.4
/nix/store/7h1kwcj29ip8vk26rhmx6bfjrap0g4l-libunwind-0.98.6
...
```

- Language features:

- @-patterns as in Haskell. For instance, in a function definition

```
f = args @ {x, y, z}: ...;
```

`args` refers to the argument as a whole, which is further pattern-matched against the attribute set pattern `{x, y, z}`.

- “...” (ellipsis) patterns. An attribute set pattern can now say ... at the end of the attribute name list to specify that the function takes *at least* the listed attributes, while ignoring additional attributes. For instance,

```
{stdenv, fetchurl, fuse, ...}: ...
```

defines a function that accepts any attribute set that includes at least the three listed attributes.

- New primops: `builtins.parseDrvName` (split a package name string like "nix-0.12pre12876" into its name and version components, e.g. "nix" and "0.12pre12876"), `builtins.compareVersions` (compare two version strings using the same algorithm that `nix-env` uses), `builtins.length` (efficiently compute the length of a list), `builtins.mul` (integer multiplication), `builtins.div` (integer division).
- `nix-prefetch-url` now supports `mirror://` URLs, provided that the environment variable `NIXPKGS_ALL` points at a Nixpkgs tree.
- Removed the commands `nix-pack-closure` and `nix-unpack-closure`. You can do almost the same thing but much more efficiently by doing `nix-store --export $(nix-store -qR paths) > closure` and `nix-store --import < closure`.
- Lots of bug fixes, including a big performance bug in the handling of `with`-expressions.

# Release 0.11 (2007-12-31)

Nix 0.11 has many improvements over the previous stable release. The most important improvement is secure multi-user support. It also features many usability enhancements and language extensions, many of them prompted by NixOS, the purely functional Linux distribution based on Nix. Here is an (incomplete) list:

- Secure multi-user support. A single Nix store can now be shared between multiple (possibly untrusted) users. This is an important feature for NixOS, where it allows non-root users to install software. The old setuid method for sharing a store between multiple users has been removed. Details for setting up a multi-user store can be found in the manual.
- The new command `nix-copy-closure` gives you an easy and efficient way to exchange software between machines. It copies the missing parts of the closure of a set of store path to or from a remote machine via `ssh`.
- A new kind of string literal: strings between double single-quotes (`''`) have indentation “intelligently” removed. This allows large strings (such as shell scripts or configuration file fragments in NixOS) to cleanly follow the indentation of the surrounding expression. It also requires much less escaping, since `''` is less common in most languages than `"`.
- `nix-env --set` modifies the current generation of a profile so that it contains exactly the specified derivation, and nothing else. For example, `nix-env -p /nix/var/nix/profiles/browser --set firefox` lets the profile named `browser` contain just Firefox.
- `nix-env` now maintains meta-information about installed packages in profiles. The meta-information is the contents of the `meta` attribute of derivations, such as `description` or `homepage`. The command `nix-env -q --xml --meta` shows all meta-information.
- `nix-env` now uses the `meta.priority` attribute of derivations to resolve filename collisions between packages. Lower priority values denote a higher priority. For instance, the GCC wrapper package and the Binutils package in Nixpkgs both have a file `bin/ld`, so previously if you tried to install both you would get a collision. Now, on the other hand, the GCC wrapper declares a higher priority than Binutils, so the former’s `bin/ld` is symlinked in the user environment.
- `nix-env -i / -u`: instead of breaking package ties by version, break them by priority and version number. That is, if there are multiple packages with the same name, then

pick the package with the highest priority, and only use the version if there are multiple packages with the same priority.

This makes it possible to mark specific versions/variant in Nixpkgs more or less desirable than others. A typical example would be a beta version of some package (e.g., `gcc-4.2.0rc1`) which should not be installed even though it is the highest version, except when it is explicitly selected (e.g., `nix-env -i gcc-4.2.0rc1`).

- `nix-env --set-flag` allows meta attributes of installed packages to be modified. There are several attributes that can be usefully modified, because they affect the behaviour of `nix-env` or the user environment build script:
  - `meta.priority` can be changed to resolve filename clashes (see above).
  - `meta.keep` can be set to `true` to prevent the package from being upgraded or replaced. Useful if you want to hang on to an older version of a package.
  - `meta.active` can be set to `false` to “disable” the package. That is, no symlinks will be generated to the files of the package, but it remains part of the profile (so it won’t be garbage-collected). Set it back to `true` to re-enable the package.
- `nix-env -q` now has a flag `--prebuilt-only` (`-b`) that causes `nix-env` to show only those derivations whose output is already in the Nix store or that can be substituted (i.e., downloaded from somewhere). In other words, it shows the packages that can be installed “quickly”, i.e., don’t need to be built from source. The `-b` flag is also available in `nix-env -i` and `nix-env -u` to filter out derivations for which no pre-built binary is available.
- The new option `--argstr` (in `nix-env`, `nix-instantiate` and `nix-build`) is like `--arg`, except that the value is a string. For example, `--argstr system i686-linux` is equivalent to `--arg system "i686-linux"` (note that `--argstr` prevents annoying quoting around shell arguments).
- `nix-store` has a new operation `--read-log (-l) paths` that shows the build log of the given paths.
- Nix now uses Berkeley DB 4.5. The database is upgraded automatically, but you should be careful not to use old versions of Nix that still use Berkeley DB 4.4.
- The option `--max-silent-time` (corresponding to the configuration setting `build-max-silent-time`) allows you to set a timeout on builds — if a build produces no output on `stdout` or `stderr` for the given number of seconds, it is terminated. This is useful for recovering automatically from builds that are stuck in an infinite loop.

- `nix-channel`: each subscribed channel is its own attribute in the top-level expression generated for the channel. This allows disambiguation (e.g. `nix-env -i -A nixpkgs_unstable.firefox`).
- The substitutes table has been removed from the database. This makes operations such as `nix-pull` and `nix-channel --update` much, much faster.
- `nix-pull` now supports bzip2-compressed manifests. This speeds up channels.
- `nix-prefetch-url` now has a limited form of caching. This is used by `nix-channel` to prevent unnecessary downloads when the channel hasn't changed.
- `nix-prefetch-url` now by default computes the SHA-256 hash of the file instead of the MD5 hash. In calls to `fetchurl` you should pass the `sha256` attribute instead of `md5`. You can pass either a hexadecimal or a base-32 encoding of the hash.
- Nix can now perform builds in an automatically generated “chroot”. This prevents a builder from accessing stuff outside of the Nix store, and thus helps ensure purity. This is an experimental feature.
- The new command `nix-store --optimise` reduces Nix store disk space usage by finding identical files in the store and hard-linking them to each other. It typically reduces the size of the store by something like 25-35%.
- `~/.nix-defexpr` can now be a directory, in which case the Nix expressions in that directory are combined into an attribute set, with the file names used as the names of the attributes. The command `nix-env --import` (which set the `~/.nix-defexpr` symlink) is removed.
- Derivations can specify the new special attribute `allowedReferences` to enforce that the references in the output of a derivation are a subset of a declared set of paths. For example, if `allowedReferences` is an empty list, then the output must not have any references. This is used in NixOS to check that generated files such as initial ramdisks for booting Linux don't have any dependencies.
- The new attribute `exportReferencesGraph` allows builders access to the references graph of their inputs. This is used in NixOS for tasks such as generating ISO-9660 images that contain a Nix store populated with the closure of certain paths.
- Fixed-output derivations (like `fetchurl`) can define the attribute `impureEnvVars` to allow external environment variables to be passed to builders. This is used in Nixpkgs to support proxy configuration, among other things.
- Several new built-in functions: `builtins.attrNames`, `builtins.filterSource`,

`builtins.isAttrs`, `builtinsisFunction`, `builtins.listToAttrs`,  
`builtins.stringLength`, `builtins.sub`, `builtins.substring`, `throw`,  
`builtins.trace`, `builtins.readFile`.

# Release 0.10.1 (2006-10-11)

This release fixes two somewhat obscure bugs that occur when evaluating Nix expressions that are stored inside the Nix store ([NIX-67](#)). These do not affect most users.

# Release 0.10 (2006-10-06)

---

## Note

This version of Nix uses Berkeley DB 4.4 instead of 4.3. The database is upgraded automatically, but you should be careful not to use old versions of Nix that still use Berkeley DB 4.3. In particular, if you use a Nix installed through Nix, you should run

```
$ nix-store --clear-substitutes
```

first.

---

---

## Warning

Also, the database schema has changed slightly to fix a performance issue (see below). When you run any Nix 0.10 command for the first time, the database will be upgraded automatically. This is irreversible.

---

- `nix-env` usability improvements:
  - An option `--compare-versions` (or `-c`) has been added to `nix-env --query` to allow you to compare installed versions of packages to available versions, or vice versa. An easy way to see if you are up to date with what's in your subscribed channels is `nix-env -qc \*`.
  - `nix-env --query` now takes as arguments a list of package names about which to show information, just like `--install`, etc.: for example, `nix-env -q gcc`. Note that to show all derivations, you need to specify `\*`.
  - `nix-env -i pkgname` will now install the highest available version of *pkgname*, rather than installing all available versions (which would probably give collisions) ([NIX-31](#)).
  - `nix-env (-i|-u) --dry-run` now shows exactly which missing paths will be built or substituted.
  - `nix-env -qa --description` shows human-readable descriptions of packages, provided that they have a `meta.description` attribute (which most packages in Nixpkgs don't have yet).

- New language features:

- Reference scanning (which happens after each build) is much faster and takes a constant amount of memory.
- String interpolation. Expressions like

```
--with-freetype2-library=" + freetype + "/lib"
```

can now be written as

```
--with-freetype2-library=${freetype}/lib"
```

You can write arbitrary expressions within  `${...}` , not just identifiers.

- Multi-line string literals.
  - String concatenations can now involve derivations, as in the example `--with-freetype2-library=" + freetype + "/lib"`. This was not previously possible because we need to register that a derivation that uses such a string is dependent on `freetype`. The evaluator now properly propagates this information. Consequently, the subpath operator (`~`) has been deprecated.
  - Default values of function arguments can now refer to other function arguments; that is, all arguments are in scope in the default values ( [NIX-45](#) ).
  - Lots of new built-in primitives, such as functions for list manipulation and integer arithmetic. See the manual for a complete list. All primops are now available in the set `builtins`, allowing one to test for the availability of primop in a backwards-compatible way.
  - Real let-expressions: `let x = ...; ... z = ...; in ...`.
- New commands `nix-pack-closure` and `nix-unpack-closure` than can be used to easily transfer a store path with all its dependencies to another machine. Very convenient whenever you have some package on your machine and you want to copy it somewhere else.
  - XML support:
    - `nix-env -q --xml` prints the installed or available packages in an XML representation for easy processing by other tools.

- `nix-instantiate --eval-only --xml` prints an XML representation of the resulting term. (The new flag `--strict` forces ‘deep’ evaluation of the result, i.e., list elements and attributes are evaluated recursively.)
  - In Nix expressions, the primop `builtins.toXML` converts a term to an XML representation. This is primarily useful for passing structured information to builders.
- You can now unambiguously specify which derivation to build or install in `nix-env`, `nix-instantiate` and `nix-build` using the `--attr` / `-A` flags, which takes an attribute name as argument. (Unlike symbolic package names such as `subversion-1.4.0`, attribute names in an attribute set are unique.) For instance, a quick way to perform a test build of a package in Nixpkgs is `nix-build pkgs/top-level/all-packages.nix -A foo`. `nix-env -q --attr` shows the attribute names corresponding to each derivation.
  - If the top-level Nix expression used by `nix-env`, `nix-instantiate` or `nix-build` evaluates to a function whose arguments all have default values, the function will be called automatically. Also, the new command-line switch `--arg name value` can be used to specify function arguments on the command line.
  - `nix-install-package --url URL` allows a package to be installed directly from the given URL.
  - Nix now works behind an HTTP proxy server; just set the standard environment variables `http_proxy`, `https_proxy`, `ftp_proxy` or `all_proxy` appropriately. Functions such as `fetchurl` in Nixpkgs also respect these variables.
  - `nix-build -o symlink` allows the symlink to the build result to be named something other than `result`.
  - Platform support:
    - Support for 64-bit platforms, provided a [suitably patched ATerm library](#) is used. Also, files larger than 2 GiB are now supported.
    - Added support for Cygwin (Windows, `i686-cygwin`), Mac OS X on Intel (`i686-darwin`) and Linux on PowerPC (`powerpc-linux`).
    - Users of SMP and multicore machines will appreciate that the number of builds to be performed in parallel can now be specified in the configuration file in the `build-max-jobs` setting.

- Garbage collector improvements:
  - Open files (such as running programs) are now used as roots of the garbage collector. This prevents programs that have been uninstalled from being garbage collected while they are still running. The script that detects these additional runtime roots (`find-runtime-roots.pl`) is inherently system-specific, but it should work on Linux and on all platforms that have the `lsof` utility.
  - `nix-store --gc` (a.k.a. `nix-collect-garbage`) prints out the number of bytes freed on standard output. `nix-store --gc --print-dead` shows how many bytes would be freed by an actual garbage collection.
  - `nix-collect-garbage -d` removes all old generations of *all* profiles before calling the actual garbage collector (`nix-store --gc`). This is an easy way to get rid of all old packages in the Nix store.
  - `nix-store` now has an operation `--delete` to delete specific paths from the Nix store. It won't delete reachable (non-garbage) paths unless `--ignore-liveness` is specified.
- Berkeley DB 4.4's process registry feature is used to recover from crashed Nix processes.
- A performance issue has been fixed with the `referer` table, which stores the inverse of the `references` table (i.e., it tells you what store paths refer to a given path). Maintaining this table could take a quadratic amount of time, as well as a quadratic amount of Berkeley DB log file space (in particular when running the garbage collector) ([NIX-23](#)).
- Nix now catches the `TERM` and `HUP` signals in addition to the `INT` signal. So you can now do a `killall nix-store` without triggering a database recovery.
- `bsdiff` updated to version 4.3.
- Substantial performance improvements in expression evaluation and `nix-env -qa`, all thanks to [Valgrind](#). Memory use has been reduced by a factor 8 or so. Big speedup by memoisation of path hashing.
- Lots of bug fixes, notably:
  - Make sure that the garbage collector can run successfully when the disk is full ([NIX-18](#)).
  - `nix-env` now locks the profile to prevent races between concurrent `nix-env`

operations on the same profile ( [NIX-7](#) ).

- Removed misleading messages from `nix-env -i` (e.g., installing `foo' followed by `uninstalling 'foo'`) ([NIX-17](#)).
- Nix source distributions are a lot smaller now since we no longer include a full copy of the Berkeley DB source distribution (but only the bits we need).
- Header files are now installed so that external programs can use the Nix libraries.

# Release 0.9.2 (2005-09-21)

This bug fix release fixes two problems on Mac OS X:

- If Nix was linked against statically linked versions of the ATerm or Berkeley DB library, there would be dynamic link errors at runtime.
- `nix-pull` and `nix-push` intermittently failed due to race conditions involving pipes and child processes with error messages such as `open2: open(GLOB(0x180b2e4), >&=9) failed: Bad file descriptor at /nix/bin/nix-pull line 77 (issue NIX-14 )`.

# Release 0.9.1 (2005-09-20)

This bug fix release addresses a problem with the ATerm library when the `--with-aterm` flag in `configure` was *not* used.

# Release 0.9 (2005-09-16)

NOTE: this version of Nix uses Berkeley DB 4.3 instead of 4.2. The database is upgraded automatically, but you should be careful not to use old versions of Nix that still use Berkeley DB 4.2. In particular, if you use a Nix installed through Nix, you should run

```
$ nix-store --clear-substitutes
```

first.

- Unpacking of patch sequences is much faster now since we no longer do redundant unpacking and repacking of intermediate paths.
- Nix now uses Berkeley DB 4.3.
- The `derivation` primitive is lazier. Attributes of dependent derivations can mutually refer to each other (as long as there are no data dependencies on the `outPath` and `drvPath` attributes computed by `derivation`).

For example, the expression `derivation attrs` now evaluates to (essentially)

```
attrs // {
 type = "derivation";
 outPath = derivation! attrs;
 drvPath = derivation! attrs;
}
```

where `derivation!` is a primop that does the actual derivation instantiation (i.e., it does what `derivation` used to do). The advantage is that it allows commands such as `nix-env -qa` and `nix-env -i` to be much faster since they no longer need to instantiate all derivations, just the `name` attribute.

Also, it allows derivations to cyclically reference each other, for example,

```
webServer = derivation {
 ...
 hostName = "svn.cs.uu.nl";
 services = [svnService];
};

svnService = derivation {
 ...
 hostName = webServer.hostName;
};
```

Previously, this would yield a black hole (infinite recursion).

- `nix-build` now defaults to using `./default.nix` if no Nix expression is specified.
- `nix-instantiate`, when applied to a Nix expression that evaluates to a function, will call the function automatically if all its arguments have defaults.
- Nix now uses libtool to build dynamic libraries. This reduces the size of executables.
- A new list concatenation operator `++`. For example, `[1 2 3] ++ [4 5 6]` evaluates to `[1 2 3 4 5 6]`.
- Some currently undocumented primops to support low-level build management using Nix (i.e., using Nix as a Make replacement). See the commit messages for [r3578](#) and [r3580](#).
- Various bug fixes and performance improvements.

# Release 0.8.1 (2005-04-13)

This is a bug fix release.

- Patch downloading was broken.
- The garbage collector would not delete paths that had references from invalid (but substitutable) paths.

# Release 0.8 (2005-04-11)

NOTE: the hashing scheme in Nix 0.8 changed (as detailed below). As a result, `nix-pull` manifests and channels built for Nix 0.7 and below will not work anymore. However, the Nix expression language has not changed, so you can still build from source. Also, existing user environments continue to work. Nix 0.8 will automatically upgrade the database schema of previous installations when it is first run.

If you get the error message

```
you have an old-style manifest `/nix/var/nix/manifests/[...]' ; please
delete it
```

you should delete previously downloaded manifests:

```
$ rm /nix/var/nix/manifests/*
```

If `nix-channel` gives the error message

```
manifest `http://catamaran.labs.cs.uu.nl/dist/nix/channels/[channel]/MANIFEST'
is too old (i.e., for Nix <= 0.7)
```

then you should unsubscribe from the offending channel (`nix-channel --remove URL`; leave out `/MANIFEST`), and subscribe to the same URL, with `channels` replaced by `channels-v3` (e.g., <http://catamaran.labs.cs.uu.nl/dist/nix/channels-v3/nixpkgs-unstable>).

Nix 0.8 has the following improvements:

- The cryptographic hashes used in store paths are now 160 bits long, but encoded in base-32 so that they are still only 32 characters long (e.g., `/nix/store/csw87wag8bqlqk7ipllbwypb14xainap-atk-1.9.0`). (This is actually a 160 bit truncation of a SHA-256 hash.)
- Big cleanups and simplifications of the basic store semantics. The notion of “closure store expressions” is gone (and so is the notion of “successors”); the file system references of a store path are now just stored in the database.

For instance, given any store path, you can query its closure:

```
$ nix-store -qR $(which firefox)
... lots of paths ...
```

Also, Nix now remembers for each store path the derivation that built it (the “deriver”):

```
$ nix-store -qR $(which firefox)
/nix/store/4b0jx7vq80l9aqcnkszxhymsf1ffa5jd-firefox-1.0.1.drv
```

So to see the build-time dependencies, you can do

```
$ nix-store -qR $(nix-store -qd $(which firefox))
```

or, in a nicer format:

```
$ nix-store -q --tree $(nix-store -qd $(which firefox))
```

File system references are also stored in reverse. For instance, you can query all paths that directly or indirectly use a certain Glibc:

```
$ nix-store -q --referrers-closure \
/nix/store/8lz9yc6zgmc0vlqmn2ipcpkj1mbi51vv-glibc-2.3.4
```

- The concept of fixed-output derivations has been formalised. Previously, functions such as `fetchurl` in Nixpkgs used a hack (namely, explicitly specifying a store path hash) to prevent changes to, say, the URL of the file from propagating upwards through the dependency graph, causing rebuilds of everything. This can now be done cleanly by specifying the `outputHash` and `outputHashAlgo` attributes. Nix itself checks that the content of the output has the specified hash. (This is important for maintaining certain invariants necessary for future work on secure shared stores.)
- One-click installation :-) It is now possible to install any top-level component in Nixpkgs directly, through the web — see, e.g., <http://catamaran.labs.cs.uu.nl/dist/nixpkgs-0.8/>. All you have to do is associate `/nix/bin/nix-install-package` with the MIME type `application/nix-package` (or the extension `.nixpkg`), and clicking on a package link will cause it to be installed, with all appropriate dependencies. If you just want to install some specific application, this is easier than subscribing to a channel.
- `nix-store -r PATHS` now builds all the derivations PATHS in parallel. Previously it did them sequentially (though exploiting possible parallelism between subderivations). This is nice for build farms.
- `nix-channel` has new operations `--list` and `--remove`.
- New ways of installing components into user environments:

- Copy from another user environment:

```
$ nix-env -i --from-profile .../other-profile firefox
```

- Install a store derivation directly (bypassing the Nix expression language entirely):

```
$ nix-env -i /nix/store/z58v41v21xd3...-aterm-2.3.1.drv
```

(This is used to implement `nix-install-package`, which is therefore immune to evolution in the Nix expression language.)

- Install an already built store path directly:

```
$ nix-env -i /nix/store/hsyj5pbn0d9i...-aterm-2.3.1
```

- Install the result of a Nix expression specified as a command-line argument:

```
$ nix-env -f .../i686-linux.nix -i -E 'x: x.firefoxWrapper'
```

The difference with the normal installation mode is that `-E` does not use the `name` attributes of derivations. Therefore, this can be used to disambiguate multiple derivations with the same name.

- A hash of the contents of a store path is now stored in the database after a successful build. This allows you to check whether store paths have been tampered with: `nix-store --verify --check-contents`.
- Implemented a concurrent garbage collector. It is now always safe to run the garbage collector, even if other Nix operations are happening simultaneously.

However, there can still be GC races if you use `nix-instantiate` and `nix-store --realise` directly to build things. To prevent races, use the `--add-root` flag of those commands.

- The garbage collector now finally deletes paths in the right order (i.e., topologically sorted under the “references” relation), thus making it safe to interrupt the collector without risking a store that violates the closure invariant.
- Likewise, the substitute mechanism now downloads files in the right order, thus preserving the closure invariant at all times.
- The result of `nix-build` is now registered as a root of the garbage collector. If the

- `./result` link is deleted, the GC root disappears automatically.
- The behaviour of the garbage collector can be changed globally by setting options in `/nix/etc/nix/nix.conf`.
  - `gc-keep-derivations` specifies whether deriver links should be followed when searching for live paths.
  - `gc-keep-outputs` specifies whether outputs of derivations should be followed when searching for live paths.
  - `env-keep-derivations` specifies whether user environments should store the paths of derivations when they are added (thus keeping the derivations alive).
- New `nix-env` query flags `--drv-path` and `--out-path`.
- `fetchurl` allows SHA-1 and SHA-256 in addition to MD5. Just specify the attribute `sha1` or `sha256` instead of `md5`.
- Manual updates.

# Release 0.7 (2005-01-12)

- Binary patching. When upgrading components using pre-built binaries (through `nix-pull` / `nix-channel`), Nix can automatically download and apply binary patches to already installed components instead of full downloads. Patching is “smart”: if there is a *sequence* of patches to an installed component, Nix will use it. Patches are currently generated automatically between Nixpkgs (pre-)releases.
- Simplifications to the substitute mechanism.
- Nix-pull now stores downloaded manifests in `/nix/var/nix/manifests`.
- Metadata on files in the Nix store is canonicalised after builds: the last-modified timestamp is set to 0 (00:00:00 1/1/1970), the mode is set to 0444 or 0555 (readable and possibly executable by all; setuid/setgid bits are dropped), and the group is set to the default. This ensures that the result of a build and an installation through a substitute is the same; and that timestamp dependencies are revealed.

# Release 0.6 (2004-11-14)

- Rewrite of the normalisation engine.
  - Multiple builds can now be performed in parallel (option `-j` ).
  - Distributed builds. Nix can now call a shell script to forward builds to Nix installations on remote machines, which may or may not be of the same platform type.
  - Option `--fallback` allows recovery from broken substitutes.
  - Option `--keep-going` causes building of other (unaffected) derivations to continue if one failed.
- Improvements to the garbage collector (i.e., it should actually work now).
- Setuid Nix installations allow a Nix store to be shared among multiple users.
- Substitute registration is much faster now.
- A utility `nix-build` to build a Nix expression and create a symlink to the result int the current directory; useful for testing Nix derivations.
- Manual updates.
- `nix-env` changes:
  - Derivations for other platforms are filtered out (which can be overridden using `--system-filter` ).
  - `--install` by default now uninstall previous derivations with the same name.
  - `--upgrade` allows upgrading to a specific version.
  - New operation `--delete-generations` to remove profile generations (necessary for effective garbage collection).
  - Nicer output (sorted, columnised).
- More sensible verbosity levels all around (builder output is now shown always, unless `-Q` is given).
- Nix expression language changes:

- New language construct: `with E1; E2` brings all attributes defined in the attribute set  $E1$  in scope in  $E2$ .
- Added a `map` function.
- Various new operators (e.g., string concatenation).
- Expression evaluation is much faster.
- An Emacs mode for editing Nix expressions (with syntax highlighting and indentation) has been added.
- Many bug fixes.

# Release 0.5 and earlier

Please refer to the Subversion commit log messages.