Nix 1.12

Eelco Dolstra

28 October 2017

NixOS

Status

- Please go forth and test!
- Performs database schema change, but Nix 1.11 is forwards compatible.
- NixOS: `nix.package = pkgs.nixUnstable`
- Elsewhere: `nix-env -iA nixpkgs.nixUnstable`

New CLI

- New command: `nix`
- Work in progress; does not yet replace `nix-env` and parts of `nix-store`.
- Experimental; syntax may still change.
- All commands should have a `--json` flag.

New CLI

- General syntax: `nix` *subcommand*
- Fully self-documenting:

```
$ nix copy --help
Usage: nix copy <FLAGS>... <INSTALLABLES>...

Summary: copy paths between Nix stores.

Flags:
      --all                 apply operation to the entire store
  -f, --file <FILE>         evaluate FILE rather than the default
      --from <STORE-URI>    URI of the source Nix store
      --no-check-sigs       do not require that paths are signed by trusted keys
      --no-recursive        apply operation to specified paths only
      --store <STORE-URI>   URI of the Nix store to use
  -s, --substitute          whether to try substitutes on the destination store (only supported by SSH)
      --to <STORE-URI>      URI of the destination Nix store

Examples:

  To copy Firefox from the local store to a binary cache in file:///tmp/cache:
  $ nix copy --to file:///tmp/cache -r $(type -p firefox)

  To copy the entire current NixOS system closure to another machine via SSH:
  $ nix copy --to ssh://server -r /run/current-system

  To copy a closure from another machine via SSH:
  $ nix copy --from ssh://server -r /nix/store/a6cnl93nk1wxnq84brbbwr6hxw9gp2w9-blender-2.79-rc2

Note: this program is EXPERIMENTAL and subject to change.
```

New CLI

- All configuration options are now supported as command line flags. So

  ```
  $ nix build --sandbox
  ```

  instead of

  ```
  $ nix build --option sandbox true
  ```

- `nix --help-config` shows all available configuration options.
- Configuration options can now be set in `~/.config/nix/nix.conf`.

`nix build`

- Replaces `nix-build`.
- Syntax: `nix build` *installables...*
- Examples:
  - `nix build nixpkgs.hello` = `nix-build '<nixpkgs>' -A hello`
  - `nix build -f .` = `nix-build`
  - `nix build -f /foo bar` = `nix build /foo -A bar`
- Has a progress indicator!

Remote builds

- Remote building is now much easier to do:

```
$ nix build nixpkgs.hello --builders 'root@mac x86_64-darwin'
[1/0/1 built] building hello-2.10 on ssh://root@mac: checking f
```

- No more need to set up `/etc/nix/machines.nix`, `NIX_BUILD_HOOK`, ...

`nix log`

- Replaces `nix-store --read-log`.
- Shows the build log for an installable.
- Will get it from binary caches if not available locally.
- Examples:
  - `nix log nixpkgs.hello`
  - `nix log $(type -p thunderbird)`

nix search

- Replacement for `nix-env -qa`.
- Has a cache!
- Example:

```
$ time nix search blender
warning: using cached results; pass ’-u’ to update the cache
Attribute name: nixpkgs.blender
Package name: blender
Version: 2.79
Description: 3D Creation/Animation/Publishing System

real    0m0.146s
```

nix path-info

- Replaces `nix-store -q`.
- 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
```

- Query info about a path in a binary cache:

```
$ nix path-info --json --store https://cache.nixos.org
```

- Show every path whose closure is bigger than 1 GB, sorted by closure size:

```
$ nix path-info --json --all -S
  | jq 'map(select(.closureSize > 1e9))
    | sort_by(.closureSize)
    | map([.path, .closureSize])'
```

Store URIs

- All commands accept `--store <uri>`.
- Not all stores support all operations.
- Available stores:
  - `local`: use `/nix/store`
  - `remote`: go via the Nix daemon
  - `/path`: use a chroot, i.e. `/path/nix/store`, `/path/nix/var`, ...
  - `https://`: HTTP binary cache
    - Replaces `download-from-binary-cache`
    - Supports HTTP/2
    - More reliable
  - `file://`: local binary cache
  - `s3://`: S3 binary cache
  - `ssh://`: Nix store accessed via SSH

nix copy

- A replacement for `nix-copy-closure`:

  ```
  $ nix copy nixpkgs.hello --to ssh://server
  $ nix copy --from ssh://server /nix/store/abcde-foo
  ```

- A replacement for `nix-push`:

  ```
  $ nix copy nixpkgs.hello --to file:///my-cache
  ```

- A replacement for ad hoc scripts to populate `cache.nixos.org`:

  ```
  $ nix copy nixpkgs.hello --to s3://nix-cache
  ```

Chroot stores

- Nice for testing or if you don’t have write access to `/nix/store` but still want to use `/nix/store` because of binary caches.

  ```
  $ nix build nixpkgs.hello --store ~/my-nix
  ```

nix run

- Replaces `nix-shell -p`.
- Support chroot stores.

```
$ nix run --store ~/my-nix
    nixpkgs.hello nixpkgs.bashInteractive

$ hello --version
hello (GNU Hello) 2.10

$ type -p hello
/nix/store/w5w4v29q...-hello-2.10/bin/hello

$ ls
bash: ls: command not found
```

nix verify

- Checks whether store paths are unmodified and are signed.
- Store paths now have signatures!
- Signatures are copied at substitution time.
- Locally built paths are marked as “ultimately trusted”, and you can sign them locally.

```
$ nix verify -r $(type -p thunderbird)
path '/nix/store/0nvlpdjl...-libpng-apng-1.6.31' is untrusted
[101 paths verified, 1 untrusted]
```

nix eval

- Replaces `nix-instantiate --eval`.

```
$ nix eval nixpkgs.hello.src.name
"hello-2.10.tar.gz"

$ nix eval --raw nixpkgs.lib.nixpkgsVersion
17.09.1535.1fdca25ee8

$ nix eval '(1 + 2)'
3
```

nix edit

- Opens the Nix expression corresponding to a Nix package.

```
$ EDITOR=ls nix edit nixpkgs.hello
/nix/store/wpcqnsln...-nixos-17.09.1535.1fdca25ee8/
nixos/pkgs/applications/misc/hello/default.nix
```

nix-build --hash

- A new build mode that
  - Builds a derivation
  - Computes the output hash
  - Renames the store path to the corresponding fixed-output location
- Intended to replace `nix-prefetch-*`.
- E.g. if `nix-repl.src` is a `fetchFromGitHub` call, then

```
$ nix-build --hash -A nix-repl.src
build produced path /nix/store/504a4k6z...-nix-repl-src
  with sha256 hash 0cjablz0...
```

nix why-depends

- Shows why a package has another package in its runtime closure.
- Useful for debugging closure size issues.

```
$ nix why-depends /run/current-system nixpkgs.glibc.dev
/nix/store/w29w92a4zzv79f7xvgay8z8wxpp7chl7-nixos-system-nixos-17.09.git.13f4079
|-- activate:
    ...rapped..WRAPPER_PATH=/nix/store/mghsf07ylkvbrggvvs6j25djq38v3b38-system-path/bin:/nix/store/mghs...
    => /nix/store/mghsf07ylkvbrggvvs6j25djq38v3b38-system-path
    |-- lib/debug/.build-id/09/cf9f1cd8074c79630af774ff66db62eb6820fe.debug
        -> /nix/store/6v5nz6bg926gxdbq2idj8j8ajznazbmh-nix-1.12pre5639_fda7b95c-debug/lib/debug/.build-id/09/cf
        => /nix/store/6v5nz6bg926gxdbq2idj8j8ajznazbmh-nix-1.12pre5639_fda7b95c-debug
        |-- lib/debug/.build-id/09/cf9f1cd8074c79630af774ff66db62eb6820fe.debug:
            ..._E_.._ZTSZ4mainEUlvE_./nix/store/v8qa2qbwrb7s8b4piygdiii7p7jpv068-gcc-6.4.0/lib/gcc/x86_64-unknow.
            => /nix/store/v8qa2qbwrb7s8b4piygdiii7p7jpv068-gcc-6.4.0
            |-- libexec/gcc/x86_64-unknown-linux-gnu/6.4.0/cc1:
                ...ux-gnu/include......./nix/store/50jw5m7lda3rylirxyly9diy55lh149z-glibc-2.25-49-dev/include.e
                => /nix/store/50jw5m7lda3rylirxyly9diy55lh149z-glibc-2.25-49-dev
```

builtins.fetchgit

- Fetches a Git repository *at evaluation time*.
- Does not need to specify a content hash.
- E.g. in a NixOS module:

```
imports =
  [ (builtins.fetchgit
       https://github.com/edolstra/dwarffs
       + "/module.nix")
  ];
```

- Git repos can also be specified on the command line:

```
$ nix build nixpkgs.hello
    -I nixpkgs=git://github.com/NixOS/nixpkgs
```

Structured derivation attributes

- You can now pass lists and attrsets to builders.
- So for example

  ```
  configureFlags =
    [ "--with-features='a b c'"
      "--enable-foobar"
    ];
  ```

  can now be a bash array.
- Enabled by setting `__structuredAttrs = true`.
- Will require major changes in Nixpkgs stdenv.

Placeholders

- Allow derivation attributes to refer to their own output paths.

```
configureFlags =
  [ "--prefix=${placeholder "out"}"
    "-includedir=${placeholder "dev"}"
  ];
```

Minor improvements

- `nix repl` is now part of Nix.

Minor improvements

- `nix repl` is now part of Nix.
- `channel://c` as a shortcut for `https://nixos.org/channels/c/nixexprs.tar.xz`.

Minor improvements

- `nix repl` is now part of Nix.
- `channel://c` as a shortcut for `https://nixos.org/channels/c/nixexprs.tar.xz`.
- Sandbox builds now use `/build` as a temporary directory; closes a class of security problems.

Minor improvements

- `nix repl` is now part of Nix.
- `channel://c` as a shortcut for `https://nixos.org/channels/c/nixexprs.tar.xz`.
- Sandbox builds now use `/build` as a temporary directory; closes a class of security problems.
- Linux sandbox provides `/bin/sh` by default.

Minor improvements

- `nix repl` is now part of Nix.
- `channel://c` as a shortcut for `https://nixos.org/channels/c/nixexprs.tar.xz`.
- Sandbox builds now use `/build` as a temporary directory; closes a class of security problems.
- Linux sandbox provides `/bin/sh` by default.
- Base-64 hashes.

Minor improvements

- `nix repl` is now part of Nix.
- `channel://c` as a shortcut for `https://nixos.org/channels/c/nixexprs.tar.xz`.
- Sandbox builds now use `/build` as a temporary directory; closes a class of security problems.
- Linux sandbox provides `/bin/sh` by default.
- Base-64 hashes.
- Automatic garbage collection.

Minor improvements

- `nix repl` is now part of Nix.
- `channel://c` as a shortcut for `https://nixos.org/channels/c/nixexprs.tar.xz`.
- Sandbox builds now use `/build` as a temporary directory; closes a class of security problems.
- Linux sandbox provides `/bin/sh` by default.
- Base-64 hashes.
- Automatic garbage collection.
- `max-jobs = auto`

Minor improvements

- `nix repl` is now part of Nix.
- `channel://c` as a shortcut for `https://nixos.org/channels/c/nixexprs.tar.xz`.
- Sandbox builds now use `/build` as a temporary directory; closes a class of security problems.
- Linux sandbox provides `/bin/sh` by default.
- Base-64 hashes.
- Automatic garbage collection.
- `max-jobs = auto`
- Binary cache signatures now required by default.