Software packaging and distribution for LHCb using Nix

Chris Burr[1], Ben Couturier[2] and Marco Clemencic[2] on behalf of the LHCb collaboration

[1]The University of Manchester [2]CERN

Requirements for HEP packaging

Production

Software must be stable for long periods (much longer than a LTS OS)
Need to reliably reprocess data for 10+ years, even reproducing the bugs!
Some dependencies will need to be updated such as XRootD

- Ideally made runtime dependencies with stable interfaces
- Reproducible builds help mitigate unexpected problems

Physics analysts

Want to use the latest and greatest software features to get the best results
But once ready environment must remain perfectly stable for minor fixes

Long-term analysis preservation

A single analysis often spans multiple years, requiring a stable stack during this time
Often want to combine new results with old analyses or update them with new data

What is Nix?

Nix[1] is a "purely functional package manager"

- Works with Linux and other unix systems (including macOS)
- Supports `i686`, `x86_64` and `arm64` (experimental) including cross-compilation
- Everything is kept in the **store directory** (default: `/nix/store`)
- Designed to support **many conflicting** software versions/configurations
- **Preexisting community** with O(14,000) package definitions

Nix has a very strong focus on:
Purity: All dependencies should be explicitly defined and build tools should not look in locations outside of Nix.
Reproducibility: Repeated builds should result in the same output, ideally bit-for-bit, even on other hosts.

[1] https://nixos.org/nix/

Defining packages in Nix

Defined using a custom functional language

- Knowledge of this is not required for most users

Packages are kept in a directory containing a **hash** of:

- **package source** via a SHA256 hash
- **build configuration**
- **each dependency's hashes** all the way to the libc

The hash **uniqueness** ensures:

- Many versions/configurations without conflicts
- **No ambiguity**: same install location iff same build

Example: Build both ROOT and XRootD with different Python and gcc versions → results in four different install directories for each package:

Main upstream repository of packages is **`nixpkgs`**[1]:

- Includes support for most build systems
- Many helper functions to minimise boilerplate
- Various "channels" for stable and unstable releases

Steps to add a new package:

- Create a file defining the source and dependencies
- Add one line to `all-packages.nix`

Default build script splits the build into phases:

`unpackPhase`
`patchPhase`
`configurePhase`
- Default: Run `./configure.sh` if present
- Dependencies can automatically override (i.e. `cmake`)

`buildPhase`
`checkPhase`
`installPhase`
`installCheckPhase`
`fixupPhase`
- Nix specific post-processing
- Stripping or split debug information
- Patching interpreter paths
- Remove runtime dependencies by simplifying the RPATH
- Automatically detect the remaining runtime dependencies
- Mostly achieved using `patchelf` (also a Nix project)

Build script is **flexible**, phases can be **easily overridden**

Automatic tweaks for languages and build systems

Total flexibility without any boilerplate

[1] https://nixos.org/nixpkgs/

Defining environments

Environments can also be defined using Nix

- Get the build environment for a package
- Make a meta package of symlinks (`buildEnv`)

Packages can easily define setup hooks

- Arbitrary shell script that is sourced automatically
- Can be used to easily add environment variables

See the **HSF packaging group's "testdrive"** for an example of using `buildEnv` to define a **deep stack**.

Full recipe for building the base "LHCb" software application

Here is a complete nix expression which allows the base application of the LHCb software stack to be built.

```
{ stdenv, fetchurl, boost, cmake, python, ninja, root, gaudi
, clhep, xercesc, cppunit, libxml2, openssl, relax, gsl, eigen, aida, graphviz
, qt5, mysql57, sqlite, hepmc, cool, coral, libgit2, pkgconfig, vdt, cpp-gsl
, oracle-instant-client, xrootd
# Data packages
, det-sqldddb, fieldmap, gen-decfiles, paramfiles, prconfig, raweventformat
, tck-hlttck, tck-l0tck }:

stdenv.mkDerivation rec {
  name = "LHCb-${version}";
  version = "v44r0";

  src = fetchurl {
    url = "https://gitlab.cern.ch/lhcb/LHCb/repository/${version}/archive.tar.gz";
    sha256 = "0h5wph3p3ha7h34byyamd1dlvb27hs5xpjbfff363y8r43dsk4pa";
  };

  buildInputs = [
    cmake ninja boost gaudi clhep xercesc cppunit libxml2 openssl relax eigen
    gsl aida graphviz qt5.qtbase mysql57 sqlite hepmc cool coral libgit2
    pkgconfig vdt cpp-gsl oracle-instant-client xrootd root
    (python.withPackages (ps: with ps; [ xenv pyqt5 lxml ]))
    det-sqldddb fieldmap gen-decfiles paramfiles prconfig
    raweventformat tck-hlttck tck-l0tck
  ];

  propagatedBuildInputs = [ python ];

  cmakeFlags = [
    "-GNinja"
    "-DMYSQL_INCLUDE_DIR=${mysql57}/include/"
    "-DGRAPHVIZ_INCLUDE_DIR=${graphviz}/include/"
    "-DCOOL_PYTHON_PATH=${cool}/python"
    "-DCORAL_PYTHON_PATH=${coral}/python"
  ];

  checkPhase = ''
    ninja test
  '';
  doCheck = true;

  postInstall = ''
    for fn in $out/lib/lib*.so; do \
      ${gaudi}/bin/listcomponents.exe $fn >> "''${fn%.so}.components"
    done
  '';

  enableParallelBuilding = true;

  meta = {
    homepage = http://lhcbdoc.web.cern.ch/lhcbdoc/lhcb/;
    description = "General purpose classes used throughout the LHCb software.";
    platforms = stdenv.lib.platforms.unix;
  };
}
```

Dependencies: Packages are defined as functions where the dependencies of the package are the arguments to the function. Default values for arguments are taken from all−packages.nix however they can easily overridden if required.

General attributes: Make a **derivation** with a **set** of name/value pairs, known as **attributes**, containing package details.

Source: The source to build the package which can be downloaded via https, ftp, git, svn, cvs and other. The **hash is as a dependency of the build to ensure reproduciblity.**

Build time dependencies: Dependencies which must be present at build time. Each package can modify the build environment to do tasks like setting environment variables.

Runtime dependencies: Runtime dependencies can be **automatically deduced** by searching for the presence each dependency's hash. Additional runtime dependencies can be specified using the attribute `propagatedBuildInputs`.

Custom attributes: Dependencies can modify the build procedure without requiring the default build script to support multiple build systems. Flags which are always required, such as setting install prefixes and RPATH are **included by default**, with custom attributes used for package specific dependencies.

Modify phases: Here build tests are enabled and the phase is overridden to run `ninja test` instead of `make check`.

Additional phases: Additional phases can be added at any point to allow arbitrary builds to be defined without explicitly repeating steps that are required for every build.

Package metadata: The meta attribute contains metadata about the build without interacting with the build environment. This often contains a description of the package, licensing information and a list of maintainers.

Testing Nix within LHCb

LHCb software stack

- Approximately 20 separate packages
- Distributed as binary releases on CVMFS

Changing the store directory

- Changed to `/cvmfs/lhcbdev.cern.ch/nix/`
- Would be an **essential feature for LHCb**

Custom Hydra instance dramatically improved the Nix experience

- Changing the store directory requires a full rebuild (slow!)
- Host on **CERN OpenStack**, back by **Postgres DBoD** instance
- Connect via **SSH** to **docker containers** on faster build machines
- Managing and scaling a "cluster" of build machines was easy

Forking `nixpkgs`

- Makes deep customisation easier
- Successfully **auto-rebasing** the fork to track upstream changes
- Hydra monitors for and **automatically builds** changed packages
- Will setup a system to push relevant changes upstream

Building LHCb reconstruction software (Brunel)

- Depends on 4 other LHCb packages
- Many external dependencies, most were **already available**
  - Some minor tweaks were needed
  - Oracle Instant Client:
    - **Licensing issues** prevent Nix from downloading
    - Had to manually import source
    - Enable builds of non-free software
- **Missing derivations:** CatBoost, COOL, CORAL, CLHEP, frontier, pacparser, RELAX, REFLEX, VDT, XRootD
  - Most were trivial to define
  - CatBoost:
    - Closed source build system that depends on glibc
    - Once identified easy to fix using `patchelf`

Providing binary caches with Hydra

Building deep stacks locally is time consuming and issue prone

Mitigate this with **binary caches**

- Static web servers serving **signed tarballs**
- Request file using the **package hash**

Hydra[1] is a **continuous build system**

- Deep integration with Nix
- Builds periodically, after every commit or for releases
- **Scalable** from a single machine to a entire cluster (via SSH)
- Can serve binaries directly or use plugins to export (e.g. S3)
- **Mitigations** for common issues (bad workers, network, ...)
- Can also provide **continuous integration**
- Also used by some GNU projects

[1] https://nixos.org/hydra

Summary

The LHCb stack can be built within Nix!
HSF packaging WG is considering Nix https://cern.ch/go/gf6G

Benefits:

✔ Environments are **exactly defined** and **reproducible**
✔ **Independent** from the host OS
✔ Hydra could replace Jenkins for **CI/CD needs**

Disadvantages:

✘ No **relocatability** but...
- Store directory can be changed to be on CVMFS
- Could use **containers** & **user namespaces** instead?

The University of Manchester

HEP Software Foundation