
# Test Driving Nix

## or first adventures in nixolandia...

Graeme Stewart (CERN EP-SFT)

2018-04-04

# Nix

Merriam-Webster: *nix* (v) to veto or reject

- Notwithstanding the curious etymology, as a package manager Nix certainly looked interesting
- From Chris's interesting presentation on 21 March what I liked most was:
  - Absolute package isolation and reproducibility
  - Fully consistent builds, from libc upwards
  - Full support for any combination of package builds, past and present (a.k.a. views)
    - Looks very interesting for our patching use case
- These are some rough and ready notes from some hours playing around with Nix
  - Probably I'm making lots of mistakes and I don't understand some things very well yet

# Playing on my laptop

- Installing is pretty easy:
  - `curl -o install-nix-2.0 https://nixos.org/nix/install`
  - `bash install-nix-2.0`
- After that all the basic commands worked fine
  - `nix-env -qa foo` - *search for packages*
  - `nix-env -i foo` - *install packages*
  - `nix-env -q` - *list installed packages*
  - `nix-env --list-generations` - *see environment/install history*
  - `nix-env --rollback` - *return to previous generation*
- Got some decent way to duplicating my brew packages + TexLive
  - However, I hit a unit test problem in the python3.6-notebook package
  - So could not get jupyter working out of the box
    - Is the OS X Darwin port very well supported?*

*Note I hit a similar snag with building ROOT on OS X with Spack

# Containerised Fun

- Baseline we decided for test driving was a CentOS7 container
- For a system like Nix this is almost irrelevant, due to the fact that the build is deep
- Playground Dockerfile is on GitLab
  - https://gitlab.cern.ch/graemes/utils/tree/master/docker/c7-nix

```
# Install nix into a base CentOS7 image

FROM centos:7
LABEL maintainer="Graeme Stewart <graeme.andrew.stewart@cern.ch>"

# First some prerequisites - namely curl to get the nix
# installer and bzip2 to unpack it
RUN yum -y install curl bzip2 sudo

# Now add an hsf user (passwd hsf), with sudo rights
RUN useradd -G wheel -p '$6$JmsT1xYM$CdWIKb6aHhqWniaanWDq1Hi9qF4pXJlmsSHrqDDjF8CsYG3w8WyTO4/.xBnG71Zx1aROZ.S.ZoR2BARuB4QBN1' hsf
COPY sudoers /etc/suoders
# Pre-create the /nix area because here we install as a single user
RUN mkdir -m 0755 /nix
RUN chown hsf /nix
USER hsf
ENV USER=hsf
ENV HOME=/home/hsf
WORKDIR /home/hsf
RUN curl -O https://nixos.org/nix/install
RUN bash < install

# Bootstrap the nix environment
ENV PATH=/home/hsf/.nix-profile/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/hsf/.local/bin:/home/hsf/bin
ENV NIX_PATH=nixpkgs=/home/hsf/.nix-defexpr/channels/nixpkgs
ENV NIX_SSL_CERT_FILE=/etc/ssl/certs/ca-bundle.crt
# Now install a few basic nix packages, as a development bootstrap
RUN nix-env -i gcc boost cmake python
```

Very few bootstrap dependencies

Default install is multi-user with a Nix deamon - requires sudo + questions, so that's hard to do in a Dockerfile. This install so for the hsf user only.

Not broken :-)

# Running

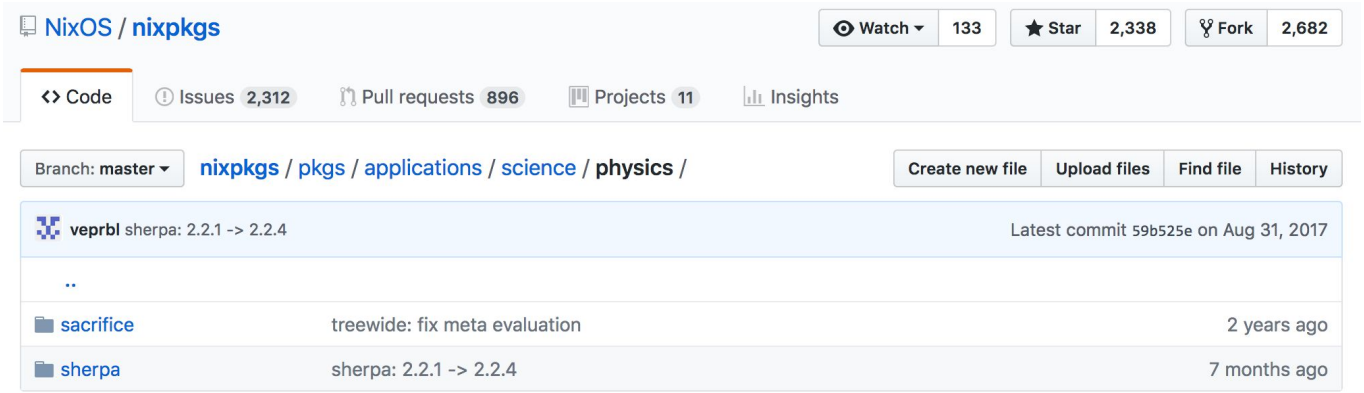
- Docker container uploaded as `graemeastewart/c7-nix`
- Start the docker container with a login shell
  - To make sure the right environment is setup

```
[teal:~$ docker run -it -v /Users/graemes:/mnt graemeastewart/c7-nix /bin/bash -l
[[hsf@5aa40b8659d6 ~]$ nix-env -q
boost-1.66_0
cmake-3.10.2
gcc-7.3.0
nix-2.0
python-2.7.14
[hsf@5aa40b8659d6 ~]$
```

- Proves again it’s not broken :-)

# Test Stack

- Default Nix collection of packages seems rather weak for scientific software:




- Clearly a lot of work needed to import our HEP packages

# Writing a build expression

- Nix packages are called *derivations*
- As Chris said, there are at least a lot of examples out there to work from and the DSL is not that hard for simple things; there is documentation
- I decided to start with a much simpler case, which is our process monitor application written for the WLCG/HSF Costs and Performance Group
- Standalone C++ application with few dependencies
  - C++11
  - CMake
  - RapidJSON ← Nix has no RapidJSON package, so start here
- What do you need?
  - An expression - DSL describing the inputs to build this package
  - A builder - just a simple script that actually does the build (usually a bash script)
  - An entry in pkgs/top-level/all-packages.nix - this concretises the function, it was the least obvious piece

# 1 - RapidJSON Expression

```
{ stdenv, fetchurl, gcc, cmake }:

stdenv.mkDerivation rec {
    name = "RapidJSON-1.1.0";
    builder = ./builder.sh;
    src = fetchurl {
        url = https://github.com/Tencent/rapidjson/archive/v1.1.0.tar.gz;
        sha256 = "bf7ced29704a1e696fbccf2a2b4ea068e7774fa37f6d7dd4039d0787f8bed98e";
    };

    doCheck = true;

    meta = {
        description = "A fast header only C++ JSON library";
        longDescription = ''
          RapidJSON is a JSON parser and generator for C++.
        '';
        homepage = http://rapidjson.org/;
        license = stdenv.lib.licenses.bsd3;

        platforms = stdenv.lib.platforms.all;
    };

  inherit gcc cmake;
}
```

# 2 - RapidJSON Builder

```
source $stdenv/setup

PATH=$gcc/bin:$cmake/bin:$PATH:/bin

echo $PATH
type ps

tar xvfz $src
mkdir build
cd build
cmake -DCMAKE_INSTALL_PERFIX=$out ../rapidjson-*
make
make install
```

- Some Nix specific boilerplate ($stdenv, $gcc, etc.)
  - Prevent accidental dependencies by being explicit about setup
- Otherwise completely standard build recipe

# 3 - Top level concretisation

- This is definitely the weirdest piece
  - The file to be modified is here:
  - /nix//store/2ws7sivczjj17yl43q3ygzycb3fz05cb-nixpkgs-18.09pre132003.13e74a838db/nixpkgs/pkgs/top-level/all-packages.nix
- However, I did find a workaround given to bootstrap a test build
  - `nix-build --keep-failed --expr 'with import <nixpkgs> {}; callPackage ./default.nix {}'`

- This is piece where I really seem to be missing something - one file for all packages that can be built?

# And then...

- Almost there, but turns out the RapidJSON code is buggy with gcc7.3
  - `-Werror=implicit-fallthrough=` catches lots of `case:` statements with questionable syntax
  - Simple to fix the code, will be an interesting exercise to see the practical steps to patch upstream code

# Summary

- Nix is quite easy to setup and get working
  - Quality of Darwin port is unclear, but Linux x86_64 seems in good shape
- Not a lot of recipes in the default repository for scientific software
- Adding expressions to build new packages is reasonably straightforward
  - Use of concretisation in `all-packages.nix` is not immediately natural
    - Maybe I am missing something here
  - There will be quite significant work to get our test stack going
- Many interesting open questions re. our use cases
  - How to globally change the compiler?
  - Or the compiler flags? (e.g., debug builds)
  - How to support multiple build flavours?
  - How to develop against software when a read-only path is used for the base, like CVMFS?
  - Hack in reolcatability (change `RUNPATH` to relative?)
- It’s early days!

# Useful links

- Dockerfile for tests
- Docker container produced (`docker pull graemeastewart/c7-nix`)
- Nix website, https://nixos.org/
- Reference manual, https://nixos.org/nixos/manual/
- Nix Pills, shorter digestible pieces of information, https://nixos.org/nixos/nix-pills/index.html