01

NixOS

PRESENTATION BY
ARPIT BHARDWAJ
21IM10009

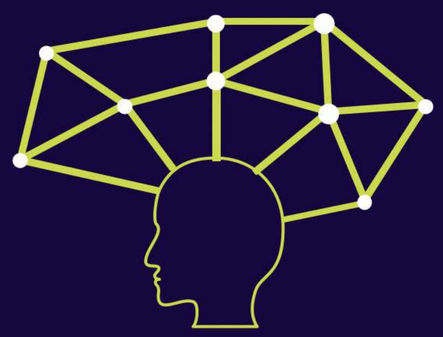
02

NIX

80,000
Nixpkgs currently maintained

10,000
Available options to choose from

WHAT EXACTLY IS IT ??



- Yet another Linux distribution?
- A programming language?
- Or just a package manager?

03

1

Nix(OS/package-manager)

Reproducible I Declarative I Reliable

04

NixOS is a Linux distribution built on top of the Nix package manager. It uses declarative configuration and allows reliable system upgrades. Several official package "channels" are offered including current Stable release and Unstable following latest development. NixOS has tools dedicated to DevOps and deployment tasks.

Nix Package manager: unlike traditional package managers, Nix is a package manager which doesn’t directly modify your system.Software which are managed by Nix are stored in a component store located in /nix/store/, instead of modifying /usr/

05

What’s special with Nix(OS/package-manager)?

- Instant development environment
- Solves many issues with current software deployment methods
- Instant runtime environment
- Easily extendible with custom modules
- Cross compilation made easy!
- Let’s you undo config changes

```
$ nixos-rebuild switch --rollback
```

And.. done.. back to previous working generation.

You just can’t break it, unless you break your booting configuration(preferably grub)
Or something like corrupting the drive, corrupting the boot-loader.. nothing can prevent
You from these things anyways ¯_(ツ)_/¯

06

Instant Development environment

1) Nix-shell

```
nix-shell -p python38 wgwt curl httpie youtube-dl ffmpeg
```

2) shell.nix

```
{ pkgs ? import <nixpkgs> {}}:

let
  # Python 3.8 environment with all of the packages I require
  pythonEnv = pkgs.python38.withPackages (ps: [
    # Scientific python essentials
    ps.numpy
    ps.scipy
    ps.pandas
    ps.matplotlib

    # Grab samples from the net via Python
    ps.requests
    ps.beautifulsoup4

    # Need this to make the samples more bearable
    ps.nltk

    # Notebook editing
    ps.jupyter
    ps.ipython
  ]);
in
pkgs.mkShell {
  packages = [
    pythonEnv        # The Python 3.8 environment I made.

    # Useful tools to grab samples from the net quickly
    pkgs.wget
    pkgs.curl
    pkgs.httpie      # I'm not good at `curl'. This might help me.

    # Download audio/video samples from the net
    pkgs.youtube-dl
    pkgs.ffmpeg
  ];
}
```

> Create an empty folder and save the snippet above as shell.nix inside it.
> Once that's done, open a terminal, navigate to the folder, and then run nix-shell.

```
$ ls
shell.nix
$ nix-shell
this derivation will be built:
  /nix/store/h3kw8zi4pi6j0p4pnrlfv4qlix0gil0d-python3-3.8.9-env.drv
these 126 paths will be fetched (100.55 MiB download, 512.38 MiB unpacked):
  /nix/store/00cxgc94bssinmw330mkf22kiggmpyhc-python3.8-nbclient-0.5.3
  ...
  /nix/store/zzcilkahrnzk9wh7902zmxgwrsks5bjs-libgcrypt-1.9.3
copying path '/nix/store/wzach25vwapl3sa6f7gylqank3jhhsin-bash-interactive-4.4-p23-dev' from 'https://cache.nixos.org'...
...
copying path '/nix/store/f7j87bxvkvs9i6z91nfnrwnd8ngn9s6q-python3.8-jupyter-1.0.0' from 'https://cache.nixos.org'...
building '/nix/store/h3kw8zi4pi6j0p4pnrlfv4qlix0gil0d-python3-3.8.9-env.drv'...
created 642 symlinks in user environment
```

> It didn't ask you to even run with sudo nor nag you about confusing missing library errors, it just works and all you need is Nix and a shell.nix file.

07

Issues with current Software Deployment method

In the Nix thesis, Eelco Dostra portrayed how ancient the current methods of software deployment is by comparing it to how software developers used to manage and allocate memory in Assembly.

- The file system mess.

  `/usr/local/nginx/` , `/opt/android/` , `/home/minecraft/`

  Solution: /usr/bin/env <name of interpreter>

- /usr/bin/env fails when specific version of interpreter is required.
  Solution: Usage of package managers, installing the particular dependency along with main package.

- Presence of other version of any dependency which don't support each other will halt the installation, Containers take uselessly high amount of storage space.

  Solution:
  - Curl | sh installer
  - Language specific package managers like npm, pip etc.
  - Usage of containers like Docker, Snap, Flatpack, AppImage etc.

- Problems with previously proposed solutions
  - Curl l sh has a very unsafe nature, as they can do literally any thing to your system.
  - Language specific installers have limited reach
  - Containers come up with everything package needs, thus are heavy on storage.

Instant runtime environment

08

- Old style Shebang

```
#!/usr/bin/env python3
```

- Nix style shebang

```
#! /usr/bin/env nix-shell
#! nix-shell -i python -p python38 python38Packages.requests python38Packages.beautifulsoup4 python38Packages.pandas
#! nix-shell -I nixpkgs=https://github.com/NixOS/nixpkgs/archive/refs/tags/21.05.tar.gz
```

This is not only limited to python, you can use it with any of your favourite language. Following is an example for bash script.

```
#! /usr/bin/env nix-shell
#! nix-shell -i bash -p gnat mktemp
#! nix-shell -I nixpkgs=https://github.com/NixOS/nixpkgs/archive/refs/tags/21.05.tar.gz
```

2

The Nix Language

Also known as Nix Expression Language

“

The Nix expression language is a

- Pure
- Lazy
- Functional

The language is not a full-featured, general purpose language. Its main job is to describe packages, compositions of packages, and the variability within packages.

Language structure

[+] Variables

```
#Comments

a = 7;
b = "string";
c = ''
    multi line string
    '';
```

[+] Sets

```
{
    a = "ehh";
    b = 1.2;
    c = {
      e = true;
      f = "oho";
    };
}
```

[+] Paths

```
./utilities/gsync-nixos
> /Users/proffapt/Desktop/utilities/gsync-nixos

<utilities/gsync-nixos>
> /Users/proffapt/Desktop/utilities/gsync-nixos
```

[+] Lists

```
a = [ 1 2 3 4 5 ];
```

[+] Functions

```
concat_a_and_b =  set: set.a + set.b
concat_a_and_b { a="hello"; b="world"; }
> "helloworld"
```

```
concat_a_and_b = {a, b}: a + b
concat_a_and_b { a="hello "; b="world"; }
> "hello world"
```

More on functions…

[-] Default arguments

```
add_a_b = { a ? 1, b ? 2 }: a + b
add_a_b {}
> 3
add_a_b {a=5;}
> 7
```

[-] Accepting unexpected arguments

```
add_a_b = { a, b }: a + b
add_a_b { a=5; b=2; c=10; }
> error: anonymous function at (string):1:2 called
> with unexpected argument 'c', at (string):1:1
add_a_b = { a, b, ... }: a + b
add_a_b { a=5; b=2; c=10; }
> 7
```

[-] Lambda Functions

```
{ a, b }:

let
a = 3;
b = 66;
in
a + b

> 69
```

[-] Storing arguments in a name

```
add_a_b = args@{ a, b, ... }: a + b + args.c
add_a_b { a=5; b=2; c=10; }
> 17
```

> Unnamed functions are called lambda functions

Language special statements

[+] With

```
pkgs.mkShell {
    packages = [
        pkgs.wget
        pkgs.curl
        pkgs.httpie

        pkgs.youtube-dl
        pkgs.ffmpeg
    ];
}
```

[+] Import

```
x = import <nixpkgs> {};
y = trace x.pkgs.hello.name x;
```

[+] Let....in

```
let
  a = 1;
  b = 2;
in  a + b
> 3
```

[+] Inherit and rec

```
buildPythonPackage rec {
    pname = "hello";
    version = "1.0";
    src = fetchPypi {
    inherit pname version;
    sha256 = "01ba..0";
    };
}
```

[+] If....else

```
new_val = if x == 7 then "yes" else "no";
```

[+] Derivations

```
derivation {
    system = "x86_64-linux";
    name = "gsync-nixos";
    builder = ./builder.sh;
    output = [ "out" ];
};
```

Better alternatives

- mkDerivation
- runCommand
- writeScriptBin

NIX REPL

```
nix-repl> :?
The following commands are available:

  <expr>        Evaluate and print expression
  <x> = <expr>  Bind expression to variable
  :a <expr>     Add attributes from resulting set to scope
  :b <expr>     Build derivation
  :e <expr>     Open package or function in $EDITOR
  :i <expr>     Build derivation, then install result into current profile
  :l <path>     Load Nix expression and add it to scope
  :lf <ref>     Load Nix flake and add it to scope
  :p <expr>     Evaluate and print expression recursively
  :q            Exit nix-repl
  :r            Reload all files
  :s <expr>     Build dependencies of derivation, then start nix-shell
  :t <expr>     Describe result of evaluation
  :u <expr>     Build derivation, then start nix-shell
  :doc <expr>   Show documentation of a builtin function
  :log <expr>   Show logs for a derivation
```

3

Creating your own Nix package

Packaging “Youtube Launcher” with nix

Looking at Youtube Launcher first

```python
#!/usr/bin/env python

from urllib.parse import urlparse

from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QLineEdit, QPushButton, QMessageBox
from PyQt5.QtCore import QProcess

app = QApplication([])

class Youtube(QWidget):
    def __init__(self):
        super().__init__()

        layout = QVBoxLayout()

        self.textbox = QLineEdit('https://www.youtube.com/watch?v=fjuJgqrZSIk')
        layout.addWidget(self.textbox)

        button = QPushButton('Watch')
        button.clicked.connect(self.__watch)
        layout.addWidget(button)

        self.setLayout(layout)

        self.setWindowTitle("YouTube Launcher");
        self.resize(512, 100)

        self.show()

    def __watch(self):
        url = urlparse(self.textbox.text())

        if (url.netloc.lower() in ('www.youtube.com', 'youtube.com') and
            url.path == '/watch' and url.query.startswith('v=') and
            len(url.query) > 2):
            QProcess().startDetached('mpv', [url.geturl()])
            app.exit()
        else:
            msg = QMessageBox()
            msg.setIcon(QMessageBox.Critical)
            msg.setText("The URL is not a valid YouTube video URL")
            msg.setWindowTitle("Error")
            msg.setStandardButtons(QMessageBox.Ok)
            msg.exec()

def main():
    window = Youtube()
    app.exec()

if __name__ == '__main__':
    main()
```

> So what are the pain points with this script?

- Written in python3.
- Uses Qt5 via PyQt5 library.
- executes mpv which requires YouTube support to be enabled.

Building the package

- To build, create an empty folder and save the snippet below as default.nix inside it.

```
let
  pkgs = import (builtins.fetchTarball {
    url = "https://github.com/NixOS/nixpkgs/archive/60cce7e5e1fdf62421ef6d4184ee399b46209366.tar.gz";
    sha256 = "100xrb925cana1kfd0c7gwkjjalq891vfgr0rn1gl9j8gp3l3gx6";
  }) {};

  mpv = pkgs.wrapMpv pkgs.mpv-unwrapped { youtubeSupport = true; };
in
pkgs.python38Packages.buildPythonApplication {
  pname = "youtube-launcher";
  version = "0.1";

  src = builtins.fetchTarball {
    url = "https://git.yukiisbo.red/yuki/youtube-launcher/-/archive/0.1/youtube-launcher-0.1.tar.gz";
    sha256 = "043sj6p47s8jk9sj9qnbshzjvxix1pfwzkpwbc5f9dqk5yxgnq13";
  };

  buildInputs = [ mpv ];
  propagatedBuildInputs = with pkgs.python38Packages; [ pyqt5 ];
  nativeBuildInputs = with pkgs.libsForQt5.qt5; [ qtbase wrapQtAppsHook ];

  postPatch = ''
    substituteInPlace youtube_launcher.py \
      --replace "'mpv'" "'${mpv}/bin/mpv'"
  '';

  preFixup = ''
    makeWrapperArgs+=("''${qtWrapperArgs[@]}")
  '';
}
```

- Now, run nix-build to build the package.

```
~ % mkdir -p experiments/youtube
~ % cd experiments/youtube
~/e/youtube % ls
~/e/youtube % nano default.nix
~/e/youtube % nix-build .
these 4 derivations will be built:
  /nix/store/vv23rn1fqfx2b7sb6n8z3qpqi9r1m5vw-builder.pl.drv
  ...
  /nix/store/5ia4ma9b96kg620k7k0w9xhgqajd828n-youtube-launcher-0.1.drv
these 335 paths will be fetched (204.35 MiB download, 984.31 MiB unpacked):
  /nix/store/03bbr1dba98kzmx364yv4dwhvssc1g1w-libxkbcommon-1.3.0
  ...
  /nix/store/zpbhr2v6x7dijfvw3w9xpbrvhp9z6gzw-hook
copying path '/nix/store/6c5x1d03c5323lsbr2il8w1gwf83cqjj-python-remove-bin-bytecode-hook' from 'https://hydra.iohk.io'...
...
copying path '/nix/store/an3gwsbssydlcwlafznv0lisq7c8hy5p-mpv-0.33.1' from 'https://cache.nixos.org'...
building '/nix/store/vv23rn1fqfx2b7sb6n8z3qpqi9r1m5vw-builder.pl.drv'...
building '/nix/store/sd5x742zvdhrm0ysx4xivyaf51qm98w7-lua-5.2.4-env.drv'...
created 12 symlinks in user environment
building '/nix/store/7imvh7yyv754jjvaqd7cgvd196xcixsj-mpv-with-scripts-0.33.1.drv'...
building '/nix/store/5ia4ma9b96kg620k7k0w9xhgqajd828n-youtube-launcher-0.1.drv'...
...
/nix/store/y1aimywh5ff57pv2azg705hlkcciy2dn-youtube-launcher-0.1
~/e/youtube %
```

- Run the package by ./result/bin/youtube-launcher inside the same folder as default.nix

```
~/e/youtube % ls -l
total 8
-rw-rw-r-- 1 yuki yuki 1001 janv. 24 13:09 default.nix
lrwxrwxrwx 1 yuki yuki   64 janv. 24 13:11 result -> /nix/store/y1aimywh5ff57pv2azg705hlkcciy2dn-youtube-launcher-0.1
~/e/youtube % ./result/bin/youtube-launcher
```

4

What is Nix doing under the hood?

Uncovering the “dark secrets” of Nix(OS/package-manager)

What does nix as an OS do ?

- **Unique hashes for the package - for coexistence of multiple version**

```
~ % ls -l /nix/store | head
total 112536
-r--r--r--  1 root root     1542 janv.  1  1970 001gp43bjqzx60cg345n2slzg7131za8-nix-nss-open-files.patch
-r--r--r--  1 root root       80 janv.  1  1970 003y7nllp3cxm5ww1q6xmygi81hhzgzn-dap-mode-recipe
-r--r--r--  1 root root     6619 janv.  1  1970 006s9a9y6iiqmsv7wm2ncm3m5lwhkv3i-libgccjit-10.3.0.drv
-r--r--r--  1 root root     3853 janv.  1  1970 007nl7dl9inal3xkca396ssxs416ggrd-python3.8-networkx-2.6.3.drv
-r--r--r--  1 root root     1670 janv.  1  1970 00nkk559kqmp7wcyq4vcvxhhk1rcjind-net-tools-2.10.drv
-r--r--r--  1 root root      437 janv.  1  1970 00qr10y7z2fcvrp9b2m46710nkjvj55z-update-autotools-gnu-config-scripts.sh
-r--r--r--  1 root root     2785 janv.  1  1970 00xlf60cvvppcn845dk30ra5kmmihspb-sqlite-simple-0.4.18.0.tar.gz.drv
-r--r--r--  1 root root    10225 janv.  1  1970 011y6iy6lh7qdwxwmm6yg3nslbxkcbkr-tdigest-0.2.1.1.drv
-r--r--r--  1 root root     1747 janv.  1  1970 0140ckl854jk2lnr65khhxjv3vwcvycj-docbook-xml-4.2.drv
```

- **Binary Caching**

Another benefit we get from Nix is binary caching. Since the store path is known before building it and we are sure that the same store path will contain the same output, we can substitute (in other words, fetch) that store path from some other location as long as that location has that path and we trust the person building it (the outputs can be signed after building so that one can store them in an untrusted place).

- **Declarative system configuration model**

```
{
    boot.loader.grub.device = "/dev/sda";

    fileSystems."/".device = "/dev/sda1";

    services.sshd.enable = true;
}
```

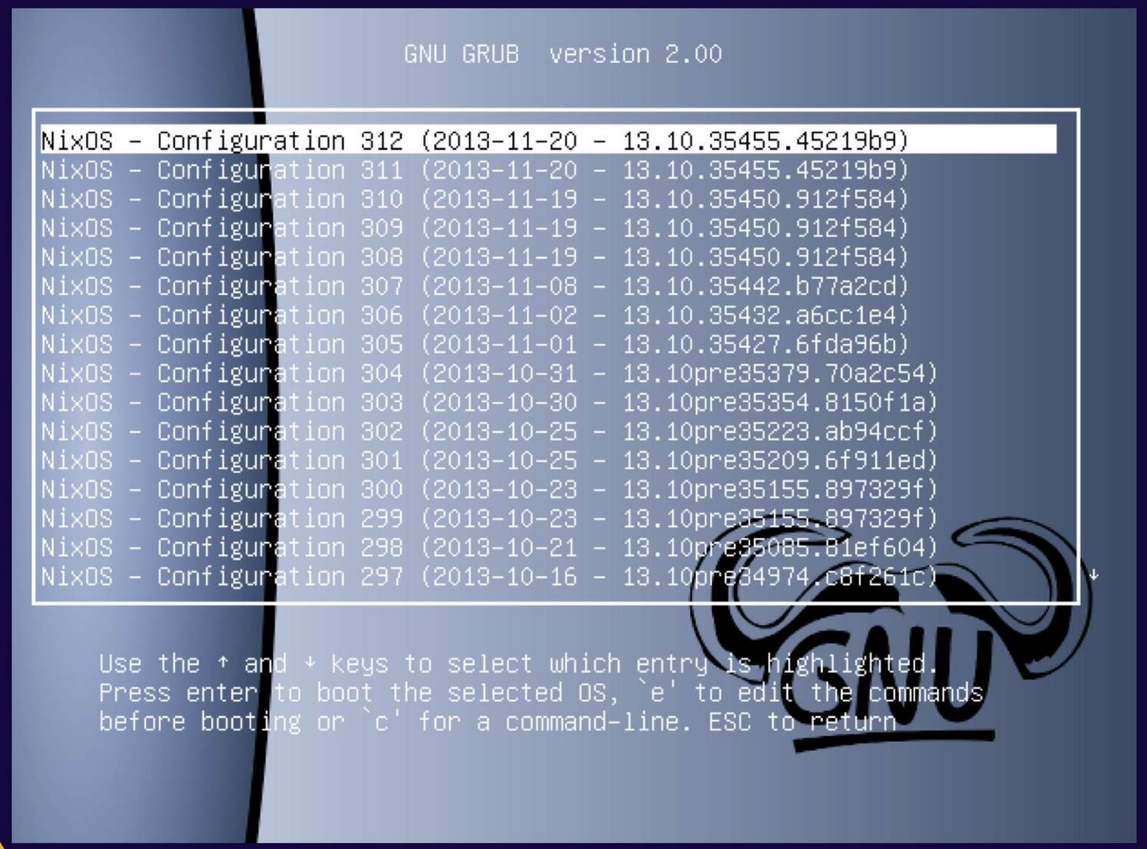
> The entire operating system — the kernel, applications, system packages, configuration files, and so on — is built by the Nix package manager from a description in nix language.

```
$ nixos-rebuild switch
```

Rollbacks

Because the files of a new configuration don’t overwrite old ones, you can (atomically) roll back to a previous configuration. For instance, if after a nixos-rebuild switch you discover that you don’t like the new configuration, you can just go back.

```
$ nixos-rebuild switch --rollback
```



> In fact, all old system configurations automatically show up in the Grub boot menu. So if the new configuration crashes or doesn’t boot properly, you can just roll back by selecting an older configuration in the Grub boot menu. In-fact, these rollbacks are Pretty fast, as they don’t require lots of files to be Restored from copies.

What does shell.nix do?

```
{ pkgs ? import <nixpkgs> {}}:                          # (1)

let
  pythonEnv = pkgs.python38.withPackages (ps: [         # (2)
```

```
in
pkgs.mkShell {                                          # (3)
  packages = [                                          # (4)
    pythonEnv
```

1) Importing Nix package collection (nixpkgs)
2) Defining variable inside let block, which will be exposed in “in” block

3) Here we’re calling the pkgs.mkShell function which allow us to make a shell environment with Nix.
4) Defining packages to be contained by shell environment. function returns the result of pkgs.mkShell

What does nix as a package manager do?

- **Instatiation**

```
~/e/youtube % nix-instantiate default.nix
/nix/store/5ia4ma9b96kg620k7k0w9xhgqajd828n-youtube-launcher-0.1.drv
```

First, Nix expressions are translated to derivations(.drv file) in a process called “instatiation”.

- **Realization**

Within the Nix store, the derivation is read by the builder which will execute the builder script and the derivation’s outputs from the build process will be stored inside the Nix store.

Looking at the package we built

```
~/e/youtube % cat ./result/bin/youtube-launcher
#! /nix/store/kzj6cblcsd1qcbbzlmbswwrn89zcmgd6-bash-4.4-p23/bin/bash -e
export PATH='/nix/store/4s0h5aawbap3xhldxhcijvl26751qrjr-python3-3.8.9/bin:/nix/store/y1aimywh5ff57pv2azg705hlkcciy2dn-yout
export PYTHONNOUSERSITE='true'
export QT_PLUGIN_PATH='/nix/store/cgpka5n7m8l31i6j9f5b071l8alqzq6v-qtbase-5.15.2-bin/lib/qt-5.15.2/plugins'${QT_PLUGIN_PATH
export QT_PLUGIN_PATH='/nix/store/k5f31g4g5s7agb7yahlsi4w7jz15516x-qtsvg-5.15.2-bin/lib/qt-5.15.2/plugins'${QT_PLUGIN_PATH:
export QT_PLUGIN_PATH='/nix/store/kv3wcr31ba5r2hglaz5d73mg084ld820-qtdeclarative-5.15.2-bin/lib/qt-5.15.2/plugins'${QT_PLUG
export QML2_IMPORT_PATH='/nix/store/kv3wcr31ba5r2hglaz5d73mg084ld820-qtdeclarative-5.15.2-bin/lib/qt-5.15.2/qml'${QML2_IMPO
export QML2_IMPORT_PATH='/nix/store/9r1hhbk3r9jv9dcnd3z59jyjs1k08r8p-qtquickcontrols-5.15.2/lib/qt-5.15.2/qml'${QML2_IMPORT
export QT_PLUGIN_PATH='/nix/store/psvnd1d97brqwgw9c8qhw4l85nlvpn6y-qtwayland-5.15.2-bin/lib/qt-5.15.2/plugins'${QT_PLUGIN_P
export QML2_IMPORT_PATH='/nix/store/psvnd1d97brqwgw9c8qhw4l85nlvpn6y-qtwayland-5.15.2-bin/lib/qt-5.15.2/qml'${QML2_IMPORT_P
exec -a "$0" "/nix/store/y1aimywh5ff57pv2azg705hlkcciy2dn-youtube-launcher-0.1/bin/.youtube-launcher-wrapped"  "$@"
```

- Wrapper which Nix generated to run the software we built

```
~/e/youtube % cat /nix/store/y1aimywh5ff57pv2azg705hlkcciy2dn-youtube-launcher-0.1/bin/.youtube-launcher-wrapped
#!/nix/store/4s0h5aawbap3xhldxhcijvl26751qrjr-python3-3.8.9/bin/python3.8
# -*- coding: utf-8 -*-
import sys;import site;import functools;sys.argv[0] = '/nix/store/y1aimywh5ff57pv2azg705hlkcciy2dn-youtube-launcher-0.1/bin
import re
import sys
from youtube_launcher import main
if __name__ == '__main__':
    sys.argv[0] = re.sub(r'(-script\.pyw|\.exe)?$', '', sys.argv[0])
    sys.exit(main())
```

- It's another wrapper written in Python generated by Nix.

```
let
  ...
in
pkgs.python38Packages.buildPythonApplication {
  ...
  nativeBuildInputs = with pkgs.libsForQt5.qt5; [ ... wrapQtAppsHook ];
  ...
  preFixup = ''
    makeWrapperArgs+=("''${qtWrapperArgs[@]}")
  '';
}
```

- What exactly are responsible for these changes?

```
~/e/youtube % cat ./result/lib/python3.8/site-packages/youtube_launcher.py
#!/usr/bin/env python

from urllib.parse import urlparse

from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QLineEdit, QPushButton, QMessageBox
from PyQt5.QtCore import QProcess


app = QApplication([])
...
QProcess().startDetached('/nix/store/p7csiywv541jnvrgah93p7zjvq8lkwkq-mpv-with-scripts-0.33.1/bin/mpv', [url.geturl()])
...
```

- Resulting built component uses deterministic paths for its runtime dependencies.

Diving deep into Instatiation

- Derivation file of our project

```
~/e/youtube % nix show-derivation /nix/store/5ia4ma9b96kg620k7k0w9xhgqajd828n-youtube-launcher-0.1.drv
{
  "/nix/store/5ia4ma9b96kg620k7k0w9xhgqajd828n-youtube-launcher-0.1.drv": {
    "outputs": {
      "out": {
        "path": "/nix/store/y1aimywh5ff57pv2azg705hlkcciy2dn-youtube-launcher-0.1"
      }
    },
    "inputSrcs": [
      "/nix/store/64fk9k7an4gyn7qgr85p42s8763yc56q-source",
      "/nix/store/9krlzvny65gdc8s7kpb6lkx8cd02c25b-default-builder.sh"
    ],
    "inputDrvs": {
      "/nix/store/2zjvqfjg2pww4lkafw2x4ni450a2w8hx-setuptools-setup-hook.drv": [
        "out"
      ],
      ...
      "/nix/store/wrnnl82wk8lz8wqgcrh8kii0kfjnl875-hook.drv": [
        "out"
      ]
    },
    "system": "x86_64-linux",
    "builder": "/nix/store/kxj6cblcsd1qcbbxlmbswwrn89zcmgd6-bash-4.4-p23/bin/bash",
    "args": [
      "-e",
      "/nix/store/9krlzvny65gdc8s7kpb6lkx8cd02c25b-default-builder.sh"
    ],
    "env": {
      "LANG": "C.UTF-8",
      "buildInputs": "/nix/store/p7csiywv541jnvrgah93p7zjvq8lkwkq-mpv-with-scripts-0.33.1",
      "builder": "/nix/store/kxj6cblcsd1qcbbxlmbswwrn89zcmgd6-bash-4.4-p23/bin/bash",
       ...
      "strictDeps": "1",
      "system": "x86_64-linux",
      "version": "0.1"
    }
  }
}
```

Env

- The Outputs(outputs)

```
"outputs": {
  "out": {
    "path": "/nix/store/y1aimywh5ff57pv2azg705hlkcciy2dn-youtube-launcher-0.1"
  }
},
```

- The Inputs(inputSrcs, inputDrvs)

```
"inputSrcs": [
  "/nix/store/64fk9k7an4gyn7qgr85p42s8763yc56q-source",
  "/nix/store/9krlzvny65gdc8s7kpb6lkx8cd02c25b-default-builder.sh"
],
```

```
"inputDrvs": {
  ...
  "/nix/store/457yxakhv0lc3df8jlb9cz0q8fk44lns-qtbase-5.15.2.drv": [
    "dev"
  ],
  "/nix/store/6pa9w933ipkki2rxxw57qpqy4ngs8q6a-python3-3.8.9.drv": [
    "out"
  ],
  ...
  "/nix/store/7imvh7yyv754jjvaqd7cgvd196xcixsj-mpv-with-scripts-0.33.1.drv": [
    "out"
  ],
  ...
  "/nix/store/jv2c1byik9bpq87rj4vzfd6p6gf9m5cj-python3.8-PyQt5-5.15.2.drv": [
    "dev"
  ],
  ...
},
```

- The Environment variables (system, builder, args, env)


```javascript
"system": "x86_64-linux",
"builder": "/nix/store/kxj6cblcsd1qcbbxlmbswwrn89zcmgd6-bash-4.4-p23/bin/bash",
"args": [
  "-e",
  "/nix/store/9krlzvny65gdc8s7kpb6lkx8cd02c25b-default-builder.sh"
],
"env": {
  "LANG": "C.UTF-8",
  "buildInputs": "/nix/store/p7csiywv541jnvrgah93p7zjvq8lkwkq-mpv-with-scripts-0.33.1",
  "builder": "/nix/store/kxj6cblcsd1qcbbxlmbswwrn89zcmgd6-bash-4.4-p23/bin/bash",
  ...
  "doInstallCheck": "1",
  "name": "youtube-launcher-0.1",
  "nativeBuildInputs": "...",
  "outputs": "out",
  "patches": "",
  "pname": "youtube-launcher",
  "postFixup": "wrapPythonPrograms\n",
  "postPatch": "substituteInPlace youtube_launcher.py \\\n  --replace \"'mpv'\" \"'/nix/store/p7csiywv541jnvrgah93p7zjvq8lk
  "preFixup": "makeWrapperArgs+=(\"${qtWrapperArgs[@]}\")\n",
  "propagatedBuildInputs": "/nix/store/cvwc3r4dcjjsa6vb50idlc4q171whvgl-python3.8-PyQt5-5.15.2-dev /nix/store/4s0h5aawbap3x
  "propagatedNativeBuildInputs": "",
  "src": "/nix/store/64fk9k7an4gyn7qgr85p42s8763yc56q-source",
  "stdenv": "/nix/store/4dlhsl4kxp9p632mbv1rcq9kjc0y6zdy-stdenv-linux",
  "strictDeps": "1",
  "system": "x86_64-linux",
  "version": "0.1"
}
```


- Have a look at this section from our package script

```
let
  ...
in
pkgs.python38Packages.buildPythonApplication {
  ...
  postPatch = ''
    substituteInPlace youtube_launcher.py \
      --replace "'mpv'" "'${mpv}/bin/mpv'"
  '';

  preFixup = ''
    makeWrapperArgs+=("''${qtWrapperArgs[@]}")
  '';
}
```

After seeing all this, you might have realised that all of that seemingly complex language ends up as a bunch environment variables configuration in the end.

Diving deep into Realization

- Args section from that same derivation file

```
"args": [
  ...
  "/nix/store/9krlzvny65gdc8s7kpb6lkx8cd02c25b-default-builder.sh"
],
```

- We see that it sources another shell script called $stdenv/setup

```
~/e/youtube % cat /nix/store/9krlzvny65gdc8s7kpb6lkx8cd02c25b-default-builder.sh
source $stdenv/setup
genericBuild
```

- Checking stdenv's value from the derivation

```
"env": {
  ...
  "stdenv": "/nix/store/4dlhsl4kxp9p632mbv1rcq9kjc0y6zdy-stdenv-linux",
  ...
}
```

```
$ ls /nix/store/4dlhsl4kxp9p632mbv1rcq9kjc0y6zdy-stdenv-linux/
nix-support  setup
$ cat /nix/store/4dlhsl4kxp9p632mbv1rcq9kjc0y6zdy-stdenv-linux/setup
export SHELL=/nix/store/kxj6cblcsd1qcbbxlmbswwrn89zcmgd6-bash-4.4-p23/bin/bash
initialPath="/nix/store/a4v1akahda85rl9gfphb07zzw79z8pb1-coreutils-8.32 ..."
defaultNativeBuildInputs="/nix/store/yayg9xvxq3f8avpvw81p7a45zqadpgvb-patchelf-0.12 ..."
defaultBuildInputs=""
# Make "strip" produce deterministic output, by setting
# timestamps etc. to a fixed value.
commonStripFlags="--enable-deterministic-archives"
export NIX_ENFORCE_PURITY="${NIX_ENFORCE_PURITY-1}"
export NIX_ENFORCE_NO_NATIVE="${NIX_ENFORCE_NO_NATIVE-1}"
NIX_LIB64_IN_SELF_RPATH=1
...
```

We get a massive bash script which a performs the build Operation. This is the "generic builder"

5

Nix Flakes

Version pinning solution

> **A flake is simply a source tree (such as a Git repository) containing a file named flake.nix that provides a standardised interface to Nix artifacts such as packages or NixOS modules. Flakes can have dependencies on other flakes, with a “lock file” pinning those dependencies to exact revisions to ensure reproducible evaluation.**

Need for Flakes

```
{ pkgs ? import <nixpkgs> {}}:
```

> Remember this ?

Issue: What if different users have different version of this channel ?

```
$ command time nix-env -qa | wc -l
5.09user 0.49system 0:05.59elapsed 99%CPU (0avgtext+0avgdata 1522792maxresident)k
28012

$ command time nix-shell --command 'exit 0'
1.34user 0.18system 0:01.69elapsed 89%CPU (0avgtext+0avgdata 434808maxresident)k
```

Issue: Slow evaluation via traditional methods

- **Getting flakes on your machine (with nix installed.. obviously)**

  1) Flakes are currently implemented in an experimental branch of Nix.
     If you want to play with flakes, you can get this version of Nix from Nixpkgs:

```
$ nix-shell -I nixpkgs=channel:nixos-21.05 --packages nixUnstable
```

  2) Since flakes are an experimental feature, you also need to add the following line to
     `~/.config/nix/nix.conf`

```
experimental-features = nix-command flakes
```

Creating your own NixFlake

```
$ nix flake metadata github:edolstra/dwarffs
Description: A filesystem that fetches DWARF debug info from the Internet on demand
…
github:edolstra/dwarffs/d11b181af08bfda367ea5cf7fad103652dc0409f
├───nix: github:NixOS/nix/3aaceeb7e2d3fb8a07a1aa5a21df1dca6bbaa0ef
|   └───nixpkgs: github:NixOS/nixpkgs/b88ff468e9850410070d4e0ccd68c7011f15b2be
└───nixpkgs: github:NixOS/nixpkgs/b88ff468e9850410070d4e0ccd68c7011f15b2be
```

> Flake metadata

- Example of flake.nix file

```
$ cat flake.nix
{
  description = "A flake for building Hello World";

  inputs.nixpkgs.url = github:NixOS/nixpkgs/nixos-20.03;

  outputs = { self, nixpkgs }: {

    defaultPackage.x86_64-linux =
      # 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";
      };

  };
}
```

```
$ nix flake show github:edolstra/dwarffs
github:edolstra/dwarffs/d11b181af08bfda367ea5cf7fad103652dc0409f
├───checks
|   ├───aarch64-linux
|   |   └───build: derivation 'dwarffs-0.1.20200409'
|   ├───i686-linux
|   |   └───build: derivation 'dwarffs-0.1.20200409'
|   └───x86_64-linux
|       └───build: derivation 'dwarffs-0.1.20200409'
├───defaultPackage
|   ├───aarch64-linux: package 'dwarffs-0.1.20200409'
|   ├───i686-linux: package 'dwarffs-0.1.20200409'
|   └───x86_64-linux: package 'dwarffs-0.1.20200409'
├───nixosModules
|   └───dwarffs: NixOS module
└───overlay: Nixpkgs overlay
```

> Flake output

Command: nix flake init also creates a basic flake.nix for you.

- Example of flake.lock file

```
$ cat flake.lock
{
  "nodes": {
    "nixpkgs": {
      "info": {
        "lastModified": 1587398327,
        "narHash": "sha256-mEKkeLgUrzAsdEaJ/1wdvYn0YZBAKEG3AN21koD2AgU="
      },
      "locked": {
        "owner": "NixOS",
        "repo": "nixpkgs",
        "rev": "5272327b81ed355bbed5659b8d303cf2979b6953",
        "type": "github"
      },
      "original": {
        "owner": "NixOS",
        "ref": "nixos-20.03",
        "repo": "nixpkgs",
        "type": "github"
      }
    },
    "root": {
      "inputs": {
        "nixpkgs": "nixpkgs"
      }
    }
  },
  "root": "root",
  "version": 5
}
```

> Untracked Files by git are invisible during Nix evaluation, in order to ensure hermetic evaluation. Thus, you need to make flake.nix visible to Git. Now build the package.

```
$ git add flake.nix
$ nix build
warning: creating lock file '/home/eelco/Dev/hello/flake.lock'
warning: Git tree '/home/eelco/Dev/hello' is dirty

$ ./result/bin/hello
Hello World
```

> In order to update the lock inputs you have to Order nix explicitly, otherwise it won't replace Existing locks. Adding new locks don't need this.

```
$ nix flake lock --update-input nixpkgs
$ nix build
```

Thanks!

Any questions?

You can find me at:

Telegram/Git(hub/lab): @proffapt
Matrix: @proffapt:matrix.org
proffapt@proton.me

Credits

Special thanks to all people who made and share these awesome resources for free:

- Write-ups by Yuki
- Blogs by tweag.io
- Videos by Matthew Croughan
- Curated list of useful links proffapt/links.md