SOLVING DUMB HACKER PROBLEMS WITH NIX

ENDING DISTRO AND PACKAGE MANAGER DRAMA FOR GOOD

(SORT OF)

HERE'S WHAT WE'RE GONNA DO

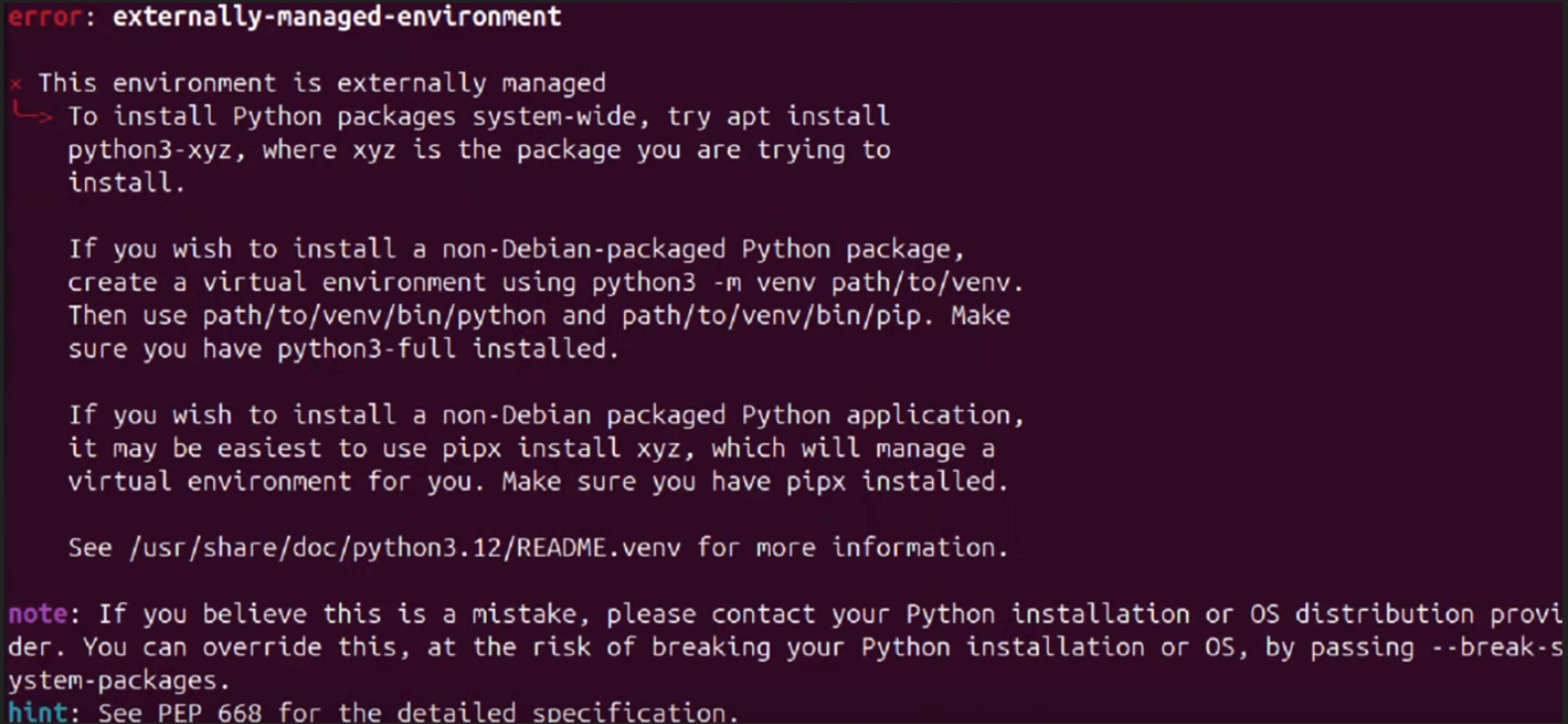
1. Talk about the problem (why does using Python suck so much?)
2. Figure out what the hell Nix is (together)
3. Solve the problem with Nix (mostly)

HOW DOES ONE USE A PYTHON-BASED TOOL?

LET US COUNT THE WAYS:

1. pip
    - I am already angry
    - dependencies? full packages?

SOMEBODY KILL ME PLEASE



```
error: externally-managed-environment

× This environment is externally managed
╰─> To install Python packages system-wide, try apt install
    python3-xyz, where xyz is the package you are trying to
    install.

    If you wish to install a non-Debian-packaged Python package,
    create a virtual environment using python3 -m venv path/to/venv.
    Then use path/to/venv/bin/python and path/to/venv/bin/pip. Make
    sure you have python3-full installed.

    If you wish to install a non-Debian packaged Python application,
    it may be easiest to use pipx install xyz, which will manage a
    virtual environment for you. Make sure you have pipx installed.

    See /usr/share/doc/python3.12/README.venv for more information.

note: If you believe this is a mistake, please contact your Python installation or OS distribution provi
der. You can override this, at the risk of breaking your Python installation or OS, by passing --break-s
ystem-packages.
hint: See PEP 668 for the detailed specification.
```

PIP INSTALL -R RAGE

1. pip
2. python-whatever?
3. virtualenv
4. pyenv
5. pipenv
6. pipx
7. conda???
8. poetry
    - a.k.a. `fuck-it-guess-im-the-developer-now`

WHY IS THIS WEIRD SNAKE MANAGING SOFTWARE

WHAT THE HELL IS NIX ANYWAY?

- Programming language?
- Operating system?
- Package manager?

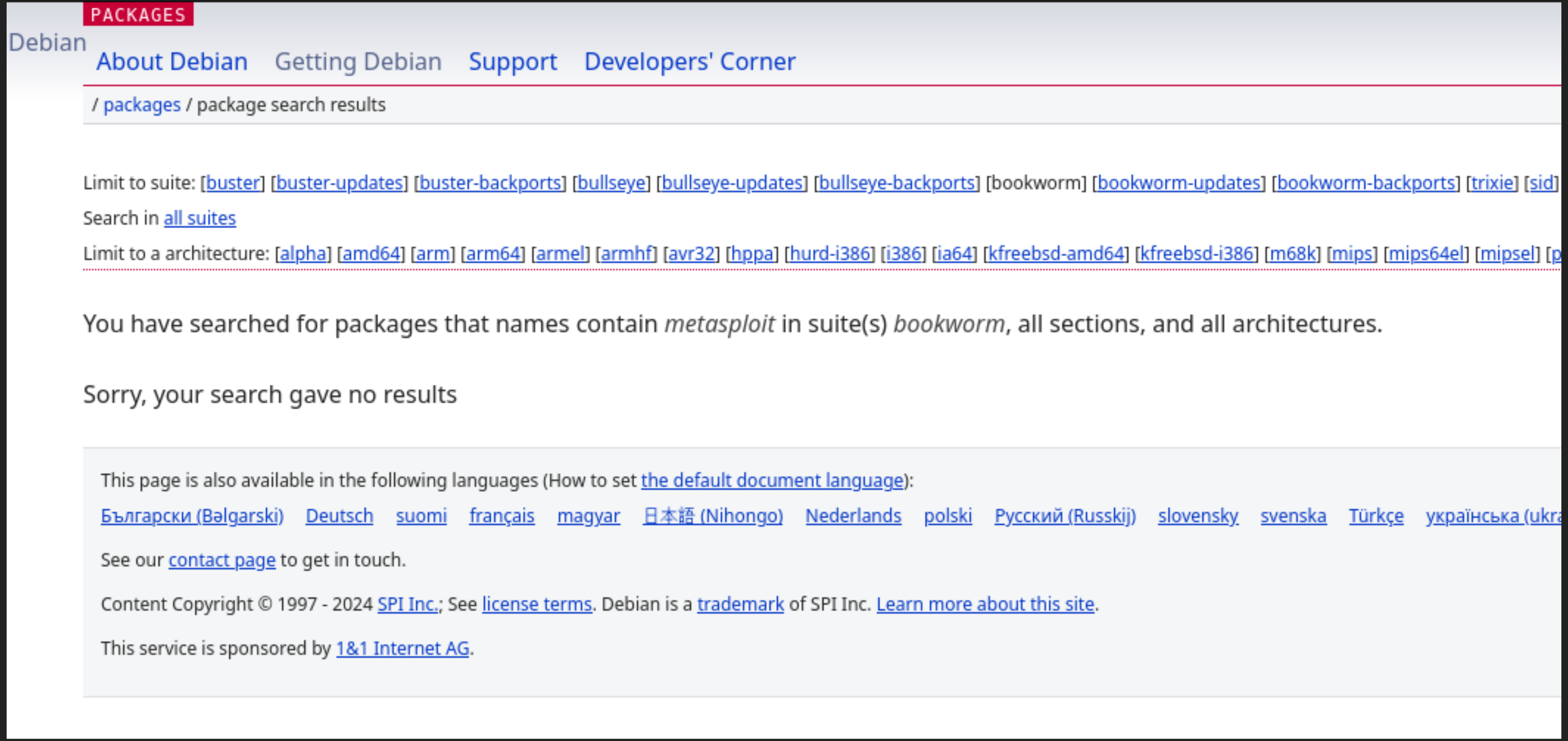
YES

BUT FOR OUR PURPOSES, IT'S A CONFIGURATION MANAGER THAT CAN INSTALL THINGS.

PROBLEM #1:

PACKAGE AVAILABILITY

(NOT) INSTALLING A PACKAGE ON DEBIAN



/ packages / package search results

Limit to suite: [buster] [buster-updates] [buster-backports] [bullseye] [bullseye-updates] [bullseye-backports] [bookworm] [bookworm-updates] [bookworm-backports] [trixie] [sid]

Search in all suites

Limit to a architecture: [alpha] [amd64] [arm] [arm64] [armel] [armhf] [avr32] [hppa] [hurd-i386] [i386] [ia64] [kfreebsd-amd64] [kfreebsd-i386] [m68k] [mips] [mips64el] [mipsel] [p

You have searched for packages that names contain *metasploit* in suite(s) *bookworm*, all sections, and all architectures.

Sorry, your search gave no results

This page is also available in the following languages (How to set the default document language):

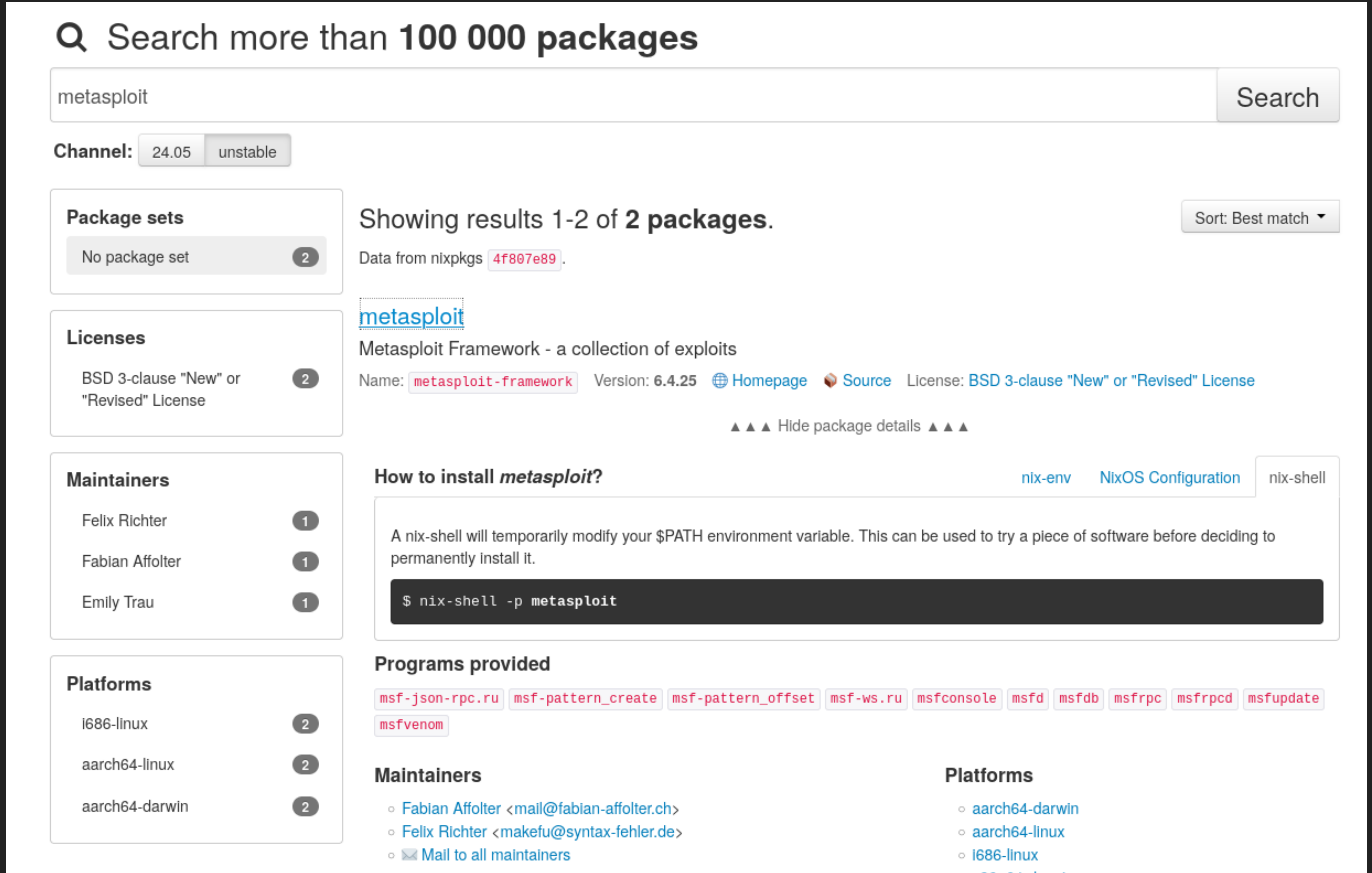
Български (Bəlgarski) Deutsch suomi français magyar 日本語 (Nihongo) Nederlands polski Русский (Russkij) slovensky svenska Türkçe українська (ukra

See our contact page to get in touch.

Content Copyright © 1997 - 2024 SPI Inc.; See license terms. Debian is a trademark of SPI Inc. Learn more about this site.

This service is sponsored by 1&1 Internet AG.

ACTUALLY INSTALLING A PACKAGE WITH NIX



Search more than **100 000 packages**

metasploit

Search

Channel: 24.05 unstable

Package sets

No package set 2

Licenses

BSD 3-clause "New" or "Revised" License 2

Maintainers

Felix Richter 1

Fabian Affolter 1

Emily Trau 1

Platforms

i686-linux 2

aarch64-linux 2

aarch64-darwin 2

Showing results 1-2 of **2 packages**.

Sort: Best match

Data from nixpkgs `4f807e89`.

metasploit

Metasploit Framework - a collection of exploits

Name: `metasploit-framework` Version: **6.4.25** Homepage Source License: BSD 3-clause "New" or "Revised" License

▲▲▲ Hide package details ▲▲▲

How to install *metasploit*?

nix-env NixOS Configuration nix-shell

A nix-shell will temporarily modify your $PATH environment variable. This can be used to try a piece of software before deciding to permanently install it.

```
$ nix-shell -p metasploit
```

Programs provided

`msf-json-rpc.ru` `msf-pattern_create` `msf-pattern_offset` `msf-ws.ru` `msfconsole` `msfd` `msfdb` `msfrpc` `msfrpcd` `msfupdate` `msfvenom`

Maintainers

- Fabian Affolter <mail@fabian-affolter.ch>
- Felix Richter <makefu@syntax-fehler.de>
- Mail to all maintainers

Platforms

- aarch64-darwin
- aarch64-linux
- i686-linux

- x86_64-darwin
- x86_64-linux

YOU CAN DO THIS MORE THAN ONE WAY

Permanent:

```
nix-env -iA metasploit
```

Temporary:

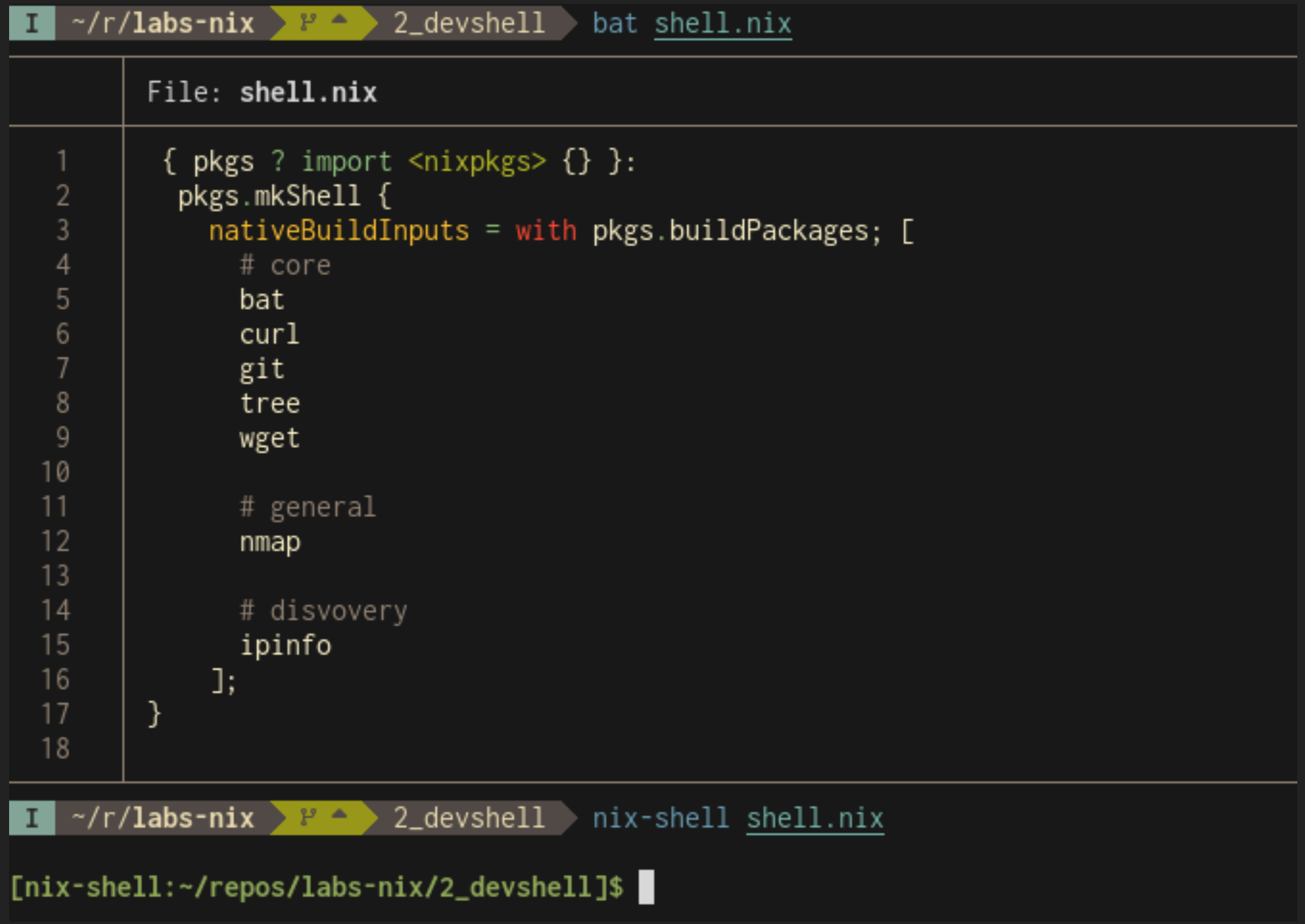
```
nix-shell -p metasploit
```

PROBLEM #2:

CONSTANT ENVIRONMENT TEARDOWN AND SETUP

DEVSHELLS

- file-based development environments
- great for moving a single set of tools around multiple systems for a quick setup



```
I  ~/r/labs-nix  2_devshell  bat shell.nix
File: shell.nix
{ pkgs ? import <nixpkgs> {} }:
 pkgs.mkShell {
   nativeBuildInputs = with pkgs.buildPackages; [
     # core
     bat
     curl
     git
     tree
     wget

     # general
     nmap

     # disvovery
     ipinfo
   ];
}

I  ~/r/labs-nix  2_devshell  nix-shell shell.nix

[nix-shell:~/repos/labs-nix/2_devshell]$
```

DIRENVS

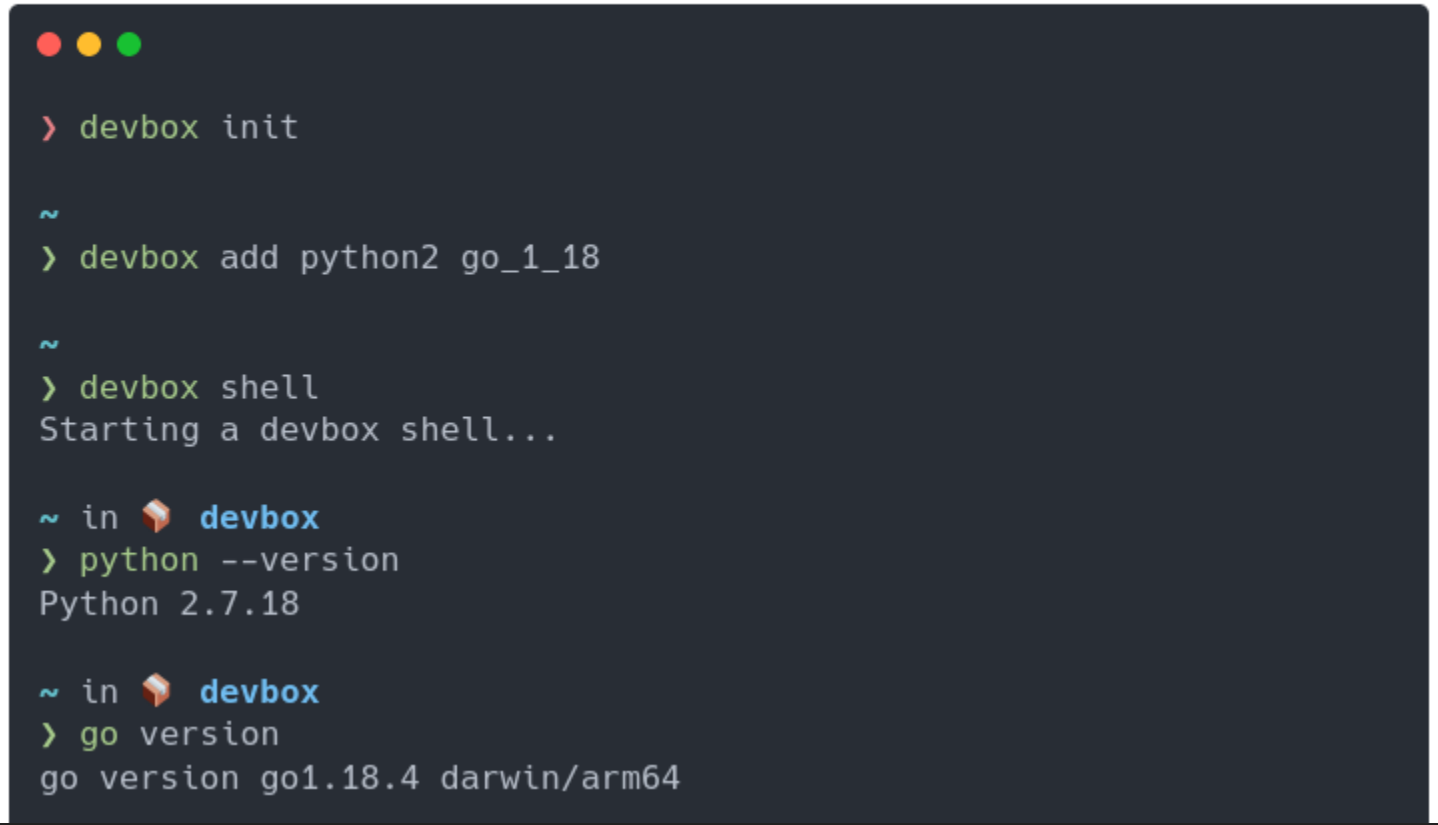
- directory-based development environments
- can handle environment variables and virtual environments easily via `.envrc`

```
# .envrc
use flake
layout python3
```

What is Devbox?

Devbox is a command-line tool that lets you easily create isolated shells for development. You start by defining the list of packages required for your project, and Devbox creates an isolated, reproducible environment with those packages installed.

In practice, Devbox works similar to a package manager like yarn – except the packages it manages are at the operating-system level (the sort of thing you would normally install with brew or apt-get).



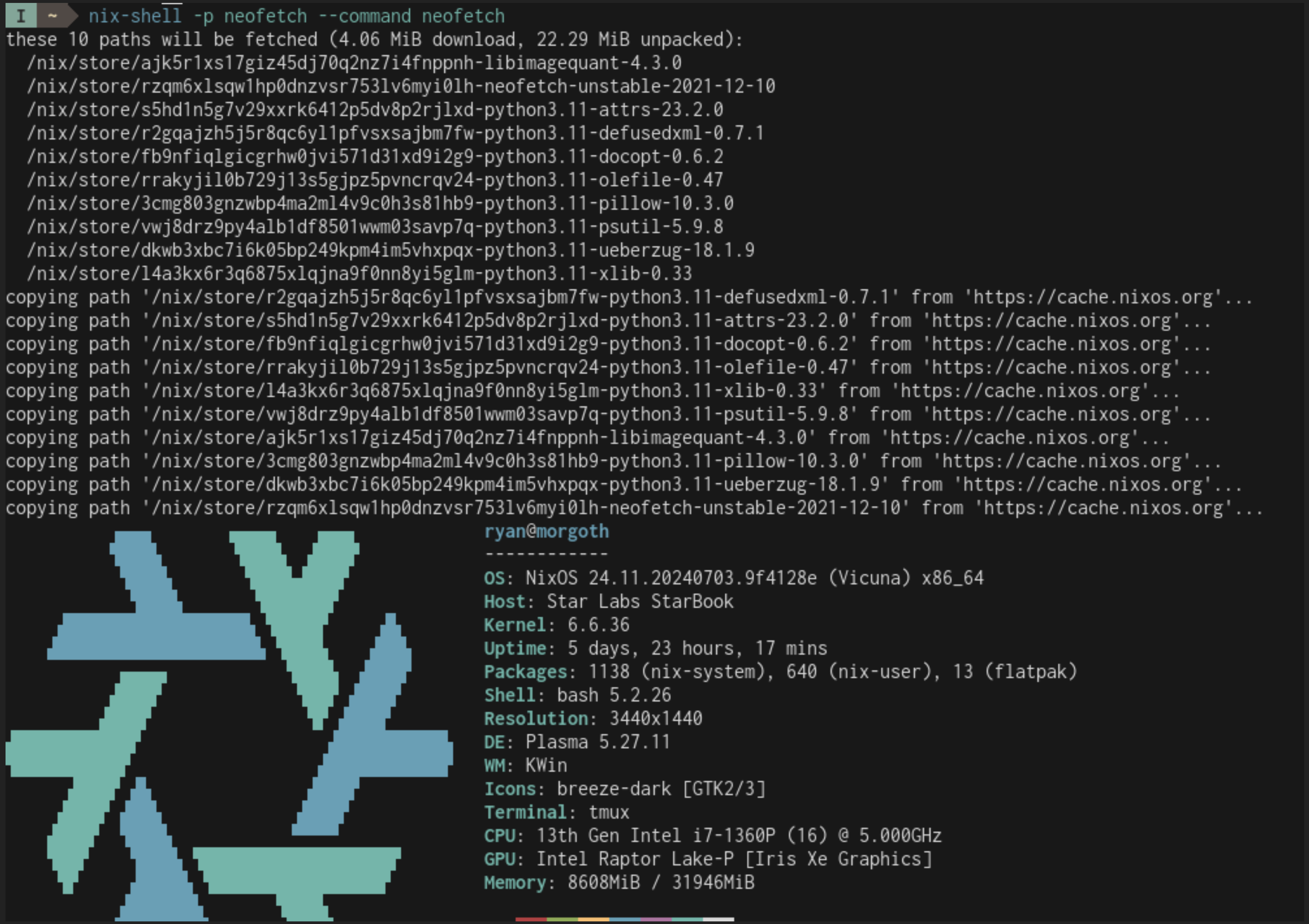
```
❯ devbox init

~
❯ devbox add python2 go_1_18

~
❯ devbox shell
Starting a devbox shell...

~ in 📦 devbox
❯ python --version
Python 2.7.18

~ in 📦 devbox
❯ go version
go version go1.18.4 darwin/arm64
```

* *I have not experimented with this but it is neat*

PROBLEM #3:

DISTROS

- here's a temporary nix-shell one-liner to prove i'm using this



```
I ~ nix-shell -p neofetch --command neofetch
these 10 paths will be fetched (4.06 MiB download, 22.29 MiB unpacked):
  /nix/store/ajk5r1xs17giz45dj70q2nz7i4fnppnh-libimagequant-4.3.0
  /nix/store/rzqm6xlsqw1hp0dnzvsr753lv6myi0lh-neofetch-unstable-2021-12-10
  /nix/store/s5hd1n5g7v29xxrk6412p5dv8p2rjlxd-python3.11-attrs-23.2.0
  /nix/store/r2gqajzh5j5r8qc6yl1pfvsxsajbm7fw-python3.11-defusedxml-0.7.1
  /nix/store/fb9nfiqlgicgrhw0jvi571d31xd9i2g9-python3.11-docopt-0.6.2
  /nix/store/rrakyjil0b729j13s5gjpz5pvncrqv24-python3.11-olefile-0.47
  /nix/store/3cmg803gnzwbp4ma2ml4v9c0h3s81hb9-python3.11-pillow-10.3.0
  /nix/store/vwj8drz9py4alb1df8501wwm03savp7q-python3.11-psutil-5.9.8
  /nix/store/dkwb3xbc7i6k05bp249kpm4im5vhxpqx-python3.11-ueberzug-18.1.9
  /nix/store/l4a3kx6r3q6875xlqjna9f0nn8yi5glm-python3.11-xlib-0.33
copying path '/nix/store/r2gqajzh5j5r8qc6yl1pfvsxsajbm7fw-python3.11-defusedxml-0.7.1' from 'https://cache.nixos.org'...
copying path '/nix/store/s5hd1n5g7v29xxrk6412p5dv8p2rjlxd-python3.11-attrs-23.2.0' from 'https://cache.nixos.org'...
copying path '/nix/store/fb9nfiqlgicgrhw0jvi571d31xd9i2g9-python3.11-docopt-0.6.2' from 'https://cache.nixos.org'...
copying path '/nix/store/rrakyjil0b729j13s5gjpz5pvncrqv24-python3.11-olefile-0.47' from 'https://cache.nixos.org'...
copying path '/nix/store/l4a3kx6r3q6875xlqjna9f0nn8yi5glm-python3.11-xlib-0.33' from 'https://cache.nixos.org'...
copying path '/nix/store/vwj8drz9py4alb1df8501wwm03savp7q-python3.11-psutil-5.9.8' from 'https://cache.nixos.org'...
copying path '/nix/store/ajk5r1xs17giz45dj70q2nz7i4fnppnh-libimagequant-4.3.0' from 'https://cache.nixos.org'...
copying path '/nix/store/3cmg803gnzwbp4ma2ml4v9c0h3s81hb9-python3.11-pillow-10.3.0' from 'https://cache.nixos.org'...
copying path '/nix/store/dkwb3xbc7i6k05bp249kpm4im5vhxpqx-python3.11-ueberzug-18.1.9' from 'https://cache.nixos.org'...
copying path '/nix/store/rzqm6xlsqw1hp0dnzvsr753lv6myi0lh-neofetch-unstable-2021-12-10' from 'https://cache.nixos.org'...
                          ryan@morgoth
                          ------------
                          OS: NixOS 24.11.20240703.9f4128e (Vicuna) x86_64
                          Host: Star Labs StarBook
                          Kernel: 6.6.36
                          Uptime: 5 days, 23 hours, 17 mins
                          Packages: 1138 (nix-system), 640 (nix-user), 13 (flatpak)
                          Shell: bash 5.2.26
                          Resolution: 3440x1440
                          DE: Plasma 5.27.11
                          WM: KWin
                          Icons: breeze-dark [GTK2/3]
                          Terminal: tmux
                          CPU: 13th Gen Intel i7-1360P (16) @ 5.000GHz
                          GPU: Intel Raptor Lake-P [Iris Xe Graphics]
                          Memory: 8608MiB / 31946MiB
```

NIXOS

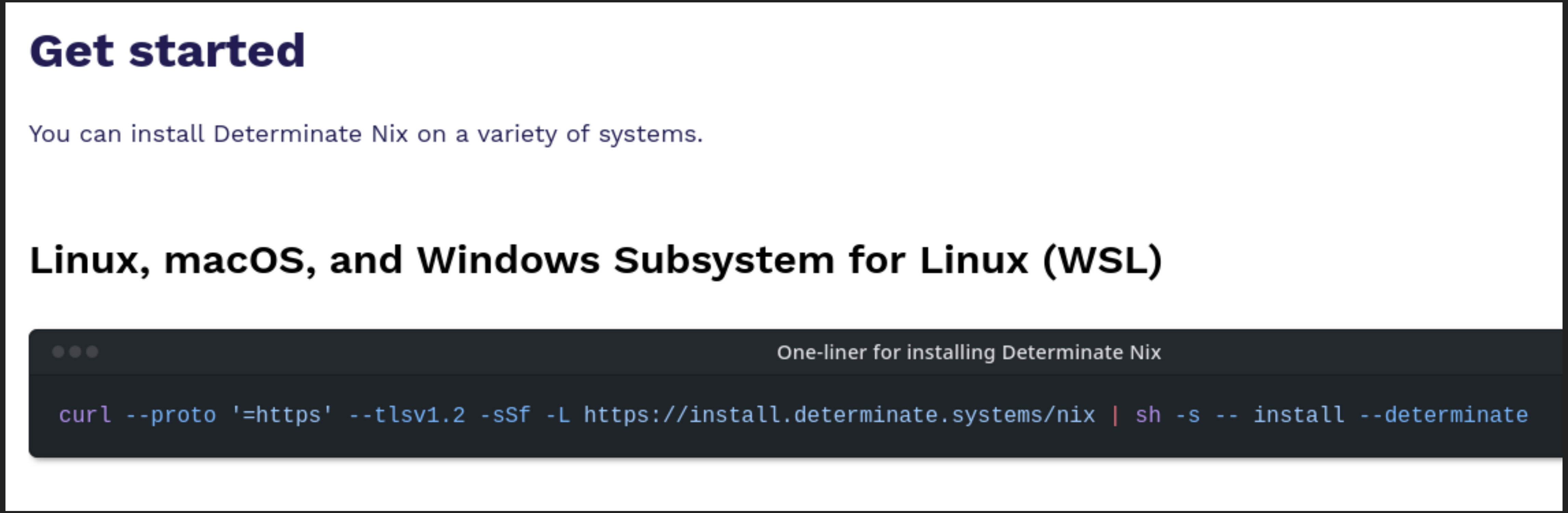
- pros
  - nix is built in
  - entire system can be configured with a few files
  - can do cool things like set `/root` as a temporary file system and nuke it on a reboot
    - `/home` stays safe
  - broke something? roll back to the last config
- cons
  - systemd
  - abstractions of abstractions
  - OpenBSD exists

MY RECOMMENDATION:

USE NIX, NOT NIXOS

Let's look at my configuration for my laptops for all the reasons.

HOW DO I TRY THIS OUT?



Get started

You can install Determinate Nix on a variety of systems.

Linux, macOS, and Windows Subsystem for Linux (WSL)

One-liner for installing Determinate Nix

```
curl --proto '=https' --tlsv1.2 -sSf -L https://install.determinate.systems/nix | sh -s -- install --determinate
```

- stores all its stuff in `/nix`, staying out of the way

At the beginning of the Zero to Nix quick start, we installed Nix using the fast and stable Determinate Nix Installer, from Determinate Systems. We hope that your journey with Nix continues well into the future, but if you need to uninstall Nix for any reason you can do so with this command:

```
/nix/nix-installer uninstall
```

Follow the prompts to approve the requested changes. Some of the changes that the installer requests:

- Delete the directory tree under `/nix`
- Delete the Nix CLI tool
- Delete all Nix-specific users and groups
- Delete the Nix configuration file at `/etc/nix/nix.conf`

Once the Determinate Nix Installer is done, you can verify that uninstallation has succeeded by confirming that directories like `/nix` and `~/.nix-profile` have been removed from your system:

```
ls /nix # error
ls ~/.nix-profile # error
```

RESOURCES

- labs from the talk: https://github.com/rybaz/labs-nix
- my nix configurations: https://github.com/rybaz/nix-conf

CONTACT

ryanbasden.com