



Solving Dumb Hacker Problems with Nix

Ending Distro and
Package Manager Drama
For Good
(sort of)

whoami --speed-date

Ryan Basden

Independent security contractor

Ex-Bishop Fox red team

Ex-risk3sixty pentesting director

OpenBSD lover, macOS cooperator

Film photography purist

Donuts are my favorite food



Important Disclaimer

You may leave this talk with a **better** understanding of Nix

Important Disclaimer

You may leave this talk with a **better** understanding of Nix

You may leave this talk with a **worse** understanding of Nix

- Yes, this is possible even if you know nothing about Nix

Important Disclaimer

You may leave this talk with a *better* understanding of Nix

You may leave this talk with a *worse* understanding of Nix

- Yes, this is possible even if you know nothing about Nix

Just know that I will leave this talk having rolled the same dice

- Please don't ask me hard questions when this is over

Sub-disclaimer

This talk is normally an hour long, so if there's time, I'll show you some labs

The labs **WILL BE AVAILABLE** on Github for you to try out yourself

Here's what we're gonna do

Here's what we're gonna do

1. Talk about why using Python sucks so much

Here's what we're gonna do

1. Talk about why using Python sucks so much
2. Figure out what the hell Nix is (together)

Here's what we're gonna do

1. Talk about why using Python sucks 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:

How does one use a python-based tool?

Let us count the ways:

1. Pip

How does one use a python-based tool?

Let us count the ways:

1. Pip

- I am already angry

`pip install unbridled-rage`

- `pip`

pip install unbridled-rage

- pip
- python-whatever?

pip install unbridled-rage

- pip
- python-whatever?
- virtualenv

pip install unbridled-rage

- pip
- python-whatever?
- virtualenv
- pyenv

pip install unbridled-rage

- pip
- python-whatever?
- virtualenv
- pyenv
- pipenv

pip install unbridled-rage

- pip
- python-whatever?
- virtualenv
- pyenv
- pipenv
- pipx

pip install unbridled-rage

- pip
- python-whatever?
- virtualenv
- pyenv
- pipenv
- pipx
- conda???

pip install unbridled-rage

- pip
- python-whatever?
- virtualenv
- pyenv
- pipenv
- pipx
- conda???
- poetry

pip install unbridled-rage

- pip
- python-whatever?
- virtualenv
- pyenv
- pipenv
- pipx
- conda???
- poetry
 - a.k.a. fuck-it-guess-im-the-developer-now

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 provider. You can override this, at the risk of breaking your Python installation or OS, by setting `PYTHONWARNINGS=ignore`.

hint: See PEP 668 for the detailed specification.

why is this weird snake managing my software



What the hell is Nix anyway?

Nix

nixpkgs

NixOS

What the hell is Nix anyway?

Nix

nixpkgs

NixOS

Package manager

- Drop-in replacement for apt, yum, pacman, brew, etc.

Can also refer to the expression language used for configuration

What the hell is Nix anyway?

Nix

Package manager

- Drop-in replacement for apt, yum, pacman, brew, etc.

Can also refer to the expression language used for configuration

nixpkgs

Package repository
natively used with the Nix
package manager

NixOS

What the hell is Nix anyway?

Nix

Package manager

- Drop-in replacement for apt, yum, pacman, brew, etc.

Can also refer to the expression language used for configuration

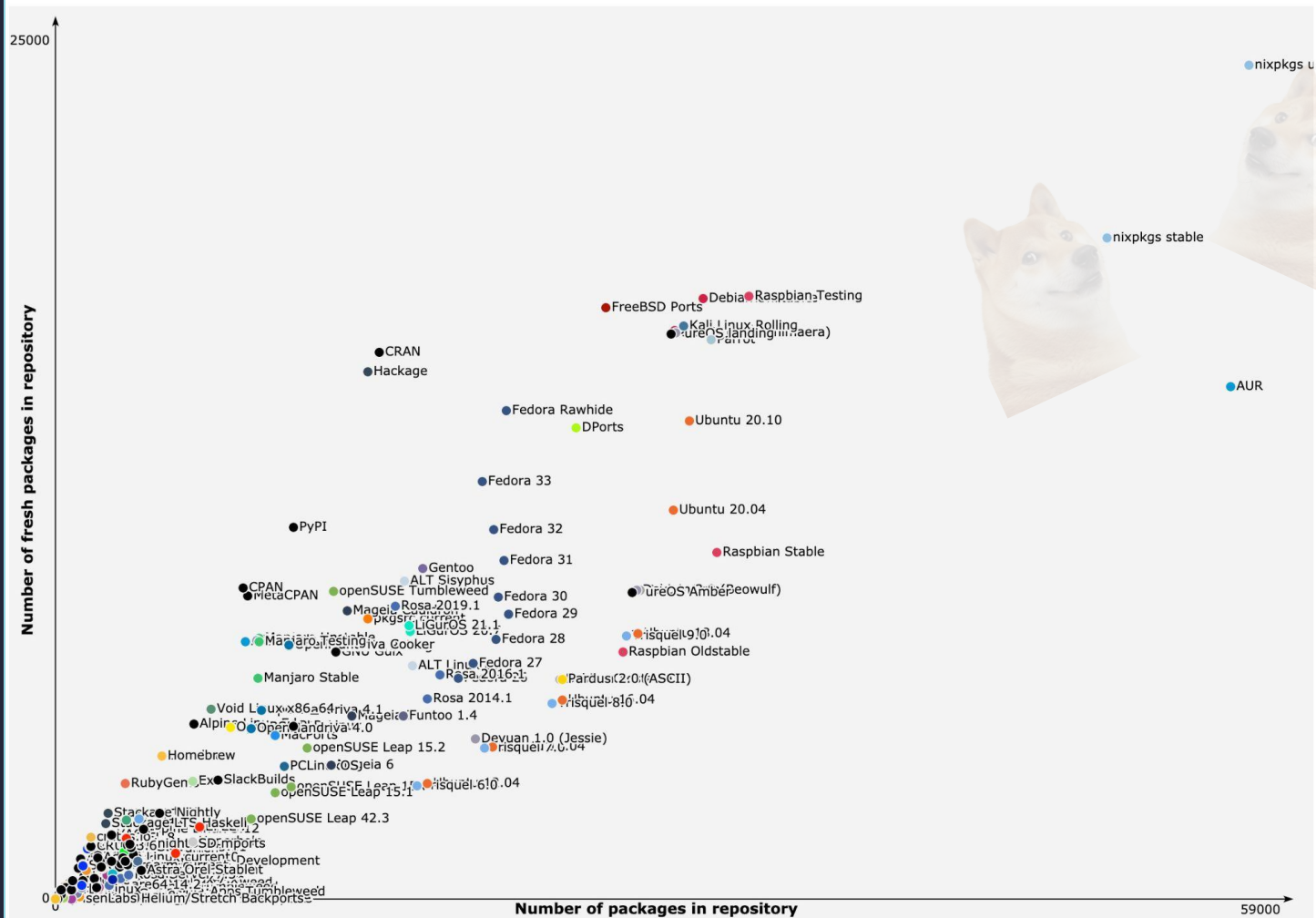
nixpkgs

Package repository
natively used with the Nix
package manager

NixOS

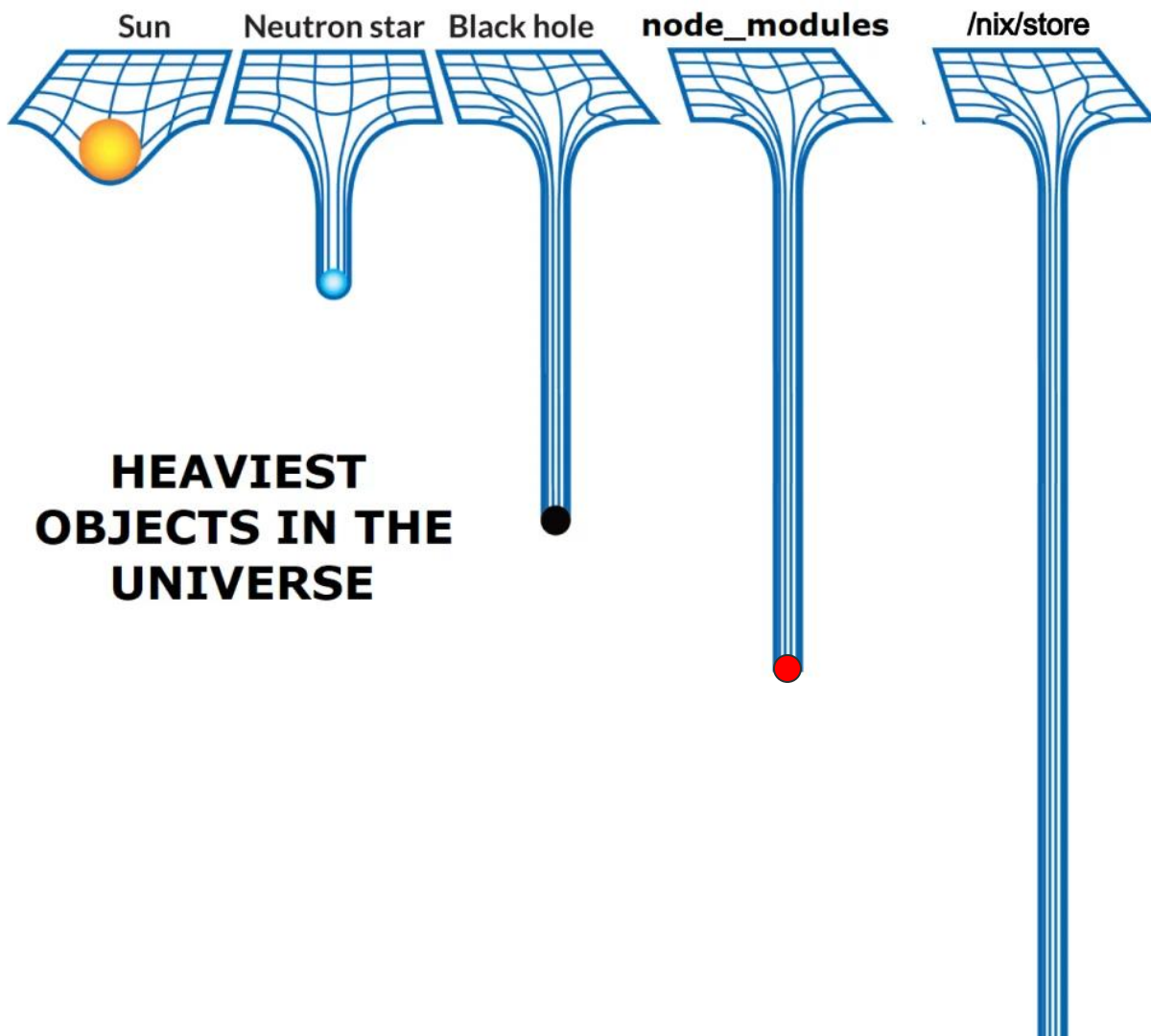
Linux distribution
specifically designed to be
completely declarative via
the Nix package
manager/language and
nixpkgs package store

Repository size/freshness map



/nix

Where the
magic
happens



drwxrwxr-x	17	root	admin	544	Oct	4	19:52	Applications/
drwxr-xr-x	67	root	wheel	2144	Sep	20	13:07	Library/
drwxr-xr-x@	10	root	wheel	320	Jul	18	23:08	System/
drwxr-xr-x	5	root	admin	160	Sep	16	19:29	Users/
drwxr-xr-x	3	root	wheel	96	Sep	25	08:34	Volumes/
drwxr-xr-x@	39	root	wheel	1248	Jul	18	23:08	bin/
drwxr-xr-x	2	root	wheel	64	Jul	18	22:08	cores/
dr-xr-xr-x	4	root	wheel	4733	Sep	24	16:39	dev/
lrwxr-xr-x@	1	root	wheel	11	Jul	18	23:08	etc@ -> private/etc
lrwxr-xr-x	1	root	wheel	25	Sep	24	16:39	home@ -> /System/Volumes/Data/home
drwxr-xr-x	9	root	wheel	288	Sep	18	18:09	nix/
drwxr-xr-x	2	root	wheel	64	Jul	18	22:08	opt/
drwxr-xr-x	6	root	wheel	192	Sep	24	16:39	private/
lrwxr-xr-x	1	root	wheel	15	Sep	24	16:39	run@ -> private/var/run
drwxr-xr-x@	64	root	wheel	2048	Jul	18	23:08	sbin/
lrwxr-xr-x@	1	root	wheel	11	Jul	18	23:08	tmp@ -> private/tmp
drwxr-xr-x@	11	root	wheel	352	Jul	18	23:08	usr/
lrwxr-xr-x@	1	root	wheel	11	Jul	18	23:08	var@ -> private/var

Are We Hackers Yet?

Tracking the availability of Kali Linux packages in NixOS

- Packages in **green** are available in the arewehackersyet tree
- Packages in **red** have been looked for and appear to be unpackaged
- Packages in **gray** haven't been checked yet

Contribute to the tracker at github.com/JJJollyjim/arewehackersyet.

Contribute to the NixOS packaging efforts at [nixpkgs#81418](https://nixpkgs.org/nixpkgs#81418).

802-11	<div><div>10</div><div>8</div></div>
base	<div><div>23</div></div>
bluetooth	<div><div>1</div><div>9</div></div>
crypto-stego	<div><div>2</div><div>4</div></div>
database	<div><div>3</div><div>7</div></div>
exploitation	<div><div>3</div><div>6</div></div>
forensics	<div><div>42</div><div>3</div><div>57</div></div>
fuzzing	<div><div>3</div><div>2</div></div>
gpu	<div><div>2</div></div>

NixOS

Pros

- Nix is built in (obviously)

NixOS

Pros

- Nix is built in (obviously)
- Entire system can be configured with a few files

NixOS

Pros

- Nix is built in (obviously)
- 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

NixOS

Pros

- Nix is built in (obviously)
- 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

NixOS

Pros

- Nix is built in (obviously)
- 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
- systemd-nspawn

NixOS

Pros

- Nix is built in (obviously)
- 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
- `systemd-nspawn`

Cons

- Abstractions of abstractions

NixOS

Pros

- Nix is built in (obviously)
- 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
- `systemd-nspawn`

Cons

- Abstractions of abstractions
- The rest of `systemd`

NixOS

Pros

- Nix is built in (obviously)
- 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
- systemd-nspawn

Cons

- Abstractions of abstractions
- The rest of systemd
- OpenBSD and macOS exist

My Recommendation

Use Nix,
NOT NixOS
(as a daily driver)

STOP USING NIXOS

- FILE NAMES WERE NOT SUPPOSED TO HAVE SHA256 HASHES
- YEARS OF EVANGELISM yet NO REAL-WORLD USE FOUND for immutability
- Want to make your system immutable anyway, for a laugh? We had a tool for that: it was called "chmod"
- "infinite recursion encountered, at undefined position" - Statement dreamed up by the utterly deranged

LOOK AT what Functional Programmers have been demanding your Respect for all this time, with all the disk space we gave them
(These are REAL packages, created by REAL Functional Programmers)

`/nix/store/pd21dgf1vdpxbfx7ilbwb8hs9l3wd6xd-binutils-wrapper-2.35.1`

`/nix/store/hbr686zx6jjsxa457w1aak0l871i232s-nixos-system-srgate-21.05pre-git`

`/nix/store/jxw9gr401dxwvzx7vb22dvxjs8gljgxr-doomed-config.json`

"Hello I would like `/nix/store/pdfnpvz6kw4yym33`

They have played us for absolute fools

```
[1]: # If you are using a Macintosh system, you may need to run `conda install cmake` first.
```

```
!pip3 install --quiet \  
  mongoengine \  
  bs4 \  
  lxml \  
  paramiko \  
  graphviz \  
  pydot-ng \  
  scapy \  
  py-postgresql \  
  psycopg2-binary \  
  Pillow \  
  pandas \  
  matplotlib \  
  scipy \  
  scikit-learn \  
  tensorflow \  
  pycryptodome \  
  cryptography \  
  ipywidgets
```

```
# Tensorflow versions < 2.16 required Mx systems to install tensorflow-macos.  
# This is no longer required.
```

error: externally-managed-environment

~~This environment is externally managed~~

↳ This command has been disabled as it tries to modify the immutable
`/nix/store` filesystem.

To use Python with Nix and nixpkgs, have a look at the online documentation:
<<https://nixos.org/manual/nixpkgs/stable/#python>>.

note: If you believe this is a mistake, please contact your Python installation or OS distribution provider. You can override this, at the risk of breaking your Python installation or OS,
by passing `--break-system-packages`.

hint: See PEP 668 for the detailed specification.

With these things out of the way, we're now ready to dive into Python!

Task 1.1

First, let's learn an important feature that makes using and troubleshooting Python using Jupyter very easy. Use the following cell to multiply 2 by 14 and include *no other*

Would you like to receive official Jupyter
news?
Please read the privacy policy.

[Open privacy policy](#)

Yes


No

[Labels](#)[Milestones](#)[New issue](#)

Maintainers leaving

⚠ Past due by 1 day 37% complete

5 Open ✓ 3 Closed

 **maintainers: remove patka** ✓ 6.topic: php 10.rebuild-darwin: 0 10.rebuild-linux: 0

#307047 opened 2 hours ago by patka-123  13 tasks


 2

 **Leave the Determinate Systems community** ✓ 6.topic: nixos 8.has: module (update) 10.rebuild-darwin: 0

10.rebuild-linux: 1-10

#307033 opened 3 hours ago by danderson


 2

 **maintainers: remove federicoschonborn** ✓ 6.topic: GNOME 6.topic: xfce 10.rebuild-darwin: 0 10.rebuild-linux: 0

#307054 opened 1 hour ago by FedericoSchonborn  1 of 13 tasks

 **maintainers: remove titanous** ✓ 10.rebuild-darwin: 0 10.rebuild-linux: 0

#307064 opened 53 minutes ago by titanous • Review required

 **remove tokudan** ● 10.rebuild-darwin: 0 10.rebuild-linux: 0

#307070 opened 33 minutes ago by tokudan  13 tasks

jtojnar



Nebucatnetzer

Jun '23

Note that there is also a bit of controversy around flakes. The feature has been implemented without going through the RFC process, lacking input from community. As a result part of the community rejected flakes.

Many also feel that it is a too big monolithic change to the Nix language, parts of which could just as well be implemented outside of Nix (see e.g. [niv](#) 50 for managing and pinning sources) and promoting one true way will stifle innovation. Having less controversial features like `nix-command` and evaluation caching tied to flakes is also considered unfortunate.

However, there is currently a new RFC to try to resolve this situation:

<https://github.com/NixOS/rfcs/pull/136> 360

3 Replies ▾



Solution

10



How to install it

Yeah, I know, just download the script and check it out to be safe

- I am not responsible for whatever happens as a result of you piping curl output to a shell

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
```

wait no i take it back

```
/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
```

Dumb Problem #1: Package Availability

PACKAGES

Debian

[About Debian](#) [Getting Debian](#) [Support](#) [Developers' Corner](#)

[/ packages](#) / package search results

Limit to suite: [buster](#) [buster-updates](#) [buster-backports](#) [bullseye](#) [bullseye-updates](#) [bullseye-backports](#) [bookworm](#) [bookworm-updates](#) [bookworm-backports](#)

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] [mips64]

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

Sorry, your search gave no results

Solution: `nixpkgs`

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`

non-NixOS package installation

permanent installation, not managed by any configuration files

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

non-NixOS package installation

permanent installation, not managed by any configuration files

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

temporary shell with package available, saved in /nix/store

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

simply exit the shell to lose access to the package

Dumb Problem #2: Local Environment Churn

Running a pentesting VM is annoying if you customize it to any degree because snapshotting can get expensive storage-wise

- Also, full VMs are kind of overkill if you're only using a CLI

Finally got an old python2 toolset (e.g. sipvicious) working and want to make sure it works again

Sometimes you get a new computer and have to set everything up again

Solution: devshells

File-based development environments

Great for moving a single set of tools around multiple systems for a quick setup

File: **shell.nix**

```
1 { pkgs ? import <nixpkgs> {} }:  
2  
3 pkgs.mkShell {  
4  
5     packages = [  
6         pkgs.python312  
7     ];  
8
```

```
$ nix-shell -p shell.nix
```

Dumb Problem #3: Dependencies

```
setuptools
six
charset_normalizer
pyasn1>=0.2.3
pyasn1_modules
pycryptodomex
pyOpenSSL==24.0.0
ldap3>=2.5,!=2.5.2,!=2.5.0,!=2.6
ldapdomaindump>=0.9.0
flask>=1.0
pyreadline3;sys_platform == 'win32'
```

Solution: devshells + flakes

```
setuptools
six
charset_normalizer
pyasn1>=0.2.3
pyasn1_modules
pycryptodomex
pyOpenSSL==24.0.0
ldap3>=2.5,!2.5.2,!2.5.0,!2.6
ldapdomaindump>=0.9.0
flask>=1.0
pyreadline3;sys_platform == 'win32'
```

\$ nix develop

```
{ pkgs ? import <nixpkgs> {} }:  
  
pkgs.mkShell {  
  
  packages = [  
    # python  
    pkgs.python312  
    # python modules  
    pkgs.python312Packages.impacket  
    pkgs.python312Packages.setuptools  
    pkgs.python312Packages.six  
    pkgs.python312Packages.charset-normalizer  
    pkgs.python312Packages.pyasn1  
    pkgs.python312Packages.pyasn1-modules  
    pkgs.python312Packages.pycryptodomex  
    pkgs.python312Packages.pyopenssl  
    pkgs.python312Packages.ldap3  
    pkgs.python312Packages.ldapdomaindump  
    pkgs.python312Packages.flask  
    # pkgs.python312Packages.pyreadline3  
  ];  
};
```

Solution: devshells + flakes

File: **flake.nix**

```
{
  description = "A very basic flake";

  inputs = {
    nixpkgs.url = "github:nixos/nixpkgs?ref=nixos-unstable";
  };

  outputs = { self, nixpkgs }:
  let
    pkgs = nixpkgs.legacyPackages."aarch64-darwin";
  in {
    devShells."aarch64-darwin".default =
      import ./shell.nix { inherit pkgs; };
  }
```

Solution: devshells + flakes

File: **flake.lock**

```
{
  "nodes": {
    "nixpkgs": {
      "locked": {
        "lastModified": 1738680400,
        "narHash": "sha256-ooLh+XW8jfa+91F1nhf90F7qhuA/y1ChLx6lXDNeY5U=",
        "owner": "nixos",
        "repo": "nixpkgs",
        "rev": "799ba5bffd04ced7067a91798353d360788b30d",
        "type": "github"
      }
    }
  }
}
```


Dumb Problem #4: Server Deployment

New VPS for every engagement

Multiple VPS deployments for more complex engagements

- C2, redirectors, etc.

No control over available AMIs or distros for deployment

80% Solution: devshells again

- Just drop a shell.nix file on your server (with Nix installed) and use it
- <https://github.com/rybaz/hackshell/blob/master/hack.nix>

```
{ pkgs ? import <nixpkgs> {} }:
  pkgs.mkShell {
    nativeBuildInputs = with pkgs.buildPackages; [

      # disvovery
      dnsx
      httpx
      ipinfo
      naabu
      subfinder

      # vulns
      nuclei
    ];
  }
```

100% Solution: colmena + NixOS

```
colmena = {  
  meta = {  
    # nixpkgs for all nodes (all Linux) by  
    nixpkgs = pkgsLinux;  
  };  
  
  # example server config  
  sliver-nix = import ./hosts/sliver-nix;
```

```
$ colmena apply --on sliver-nix
```

File: **hosts/sliver-nix/default.nix**

```
# hosts/sliver-nix/default.nix  
  
{ config, pkgs, lib, name, ... }:  
  
{  
  imports = [  
    # as well as a package specific to this node  
    ./sliver.nix  
  ];  
  
  time.timeZone = "America/New York";  
  
  # this will set the hostname to "nixie"  
  networking.hostName = name;  
  
  deployment = {  
    targetHost = "{{ SERVER IP }}";  
    targetPort = 22;  
    targetUser = "ubuntu";  
    buildOnTarget = true;  
  };  
  boot.isContainer = true;  
  time.timeZone = "America/New York";  
};  
}
```

Dumb problem #5: Isolation

More than one client at a time (hope not, but shit happens)

Docker Desktop is too resource heavy

VMware was purchased by Broadcom and now you can't download it

Solution: NixOS Containers via systemd-nspawn

Systemd can create isolated containers

PROS

- Completely native, no Docker/LXD required
- Configurable with Nix (packages, system configs like hostnames, services)

CONS

- Seccomp makes the container a tiny bit slower
- You have to be running on NixOS (not optional)

Solution: NixOS Containers via systemd-nspawn

Systemd can create isolated containers

PROS

- Completely native, no Docker/LXD required
- Configurable with Nix (packages, system configs like hostnames, services)

CONS

- Seccomp makes the container a tiny bit slower
- You have to be running on NixOS (not optional)
- You have to use systemd

Solution: NixOS Containers via systemd-nspawn

<https://github.com/redcode-labs/RedNix>

```
$ nixos-container create --flake github:redcode-labs/rednix#<arch> rednix
```

```
$ nixos-container start rednix
```

```
$ nixos-container root-login rednix
```



Resources

Labs from the talk: <https://github.com/rybaz/nix-labs/tree/wwhf>

My nix configurations:

- <https://github.com/rybaz/nix-conf>
- <https://github.com/rybaz/nix-darwin-conf> (private for now)

Contact

LinkedIn: <https://www.linkedin.com/in/ryan-basden/>

Twitter: @_rybaz

Bluesky: @rybaz.bsky.social