Demystifying nix-bitcoin

Leveraging NixOS for Secure Bitcoin Nodes

nixbitcoin.org

Jonas Nick (n1ckler)

nix-bitcoin is a collection of Nix packages and NixOS modules for easily installing **full-featured Bitcoin nodes** with an emphasis on **security**.

nix-bitcoin can be used for personal or merchant wallets, public infrastructure or for Bitcoin application backends.

Russell O’Connor

```
commit 18dc2304c069b1d2b13615cf83b919a16d1edc69
Author: Jonas Nick <jonasd.nick@gmail.com>
Date:   Tue Nov 13 23:44:54 2018 +0000

    running bitcoin
```

[@nixbitcoindev]

[@erikarvstedt]

nix-bitcoin Retweeted

Siim @siim · Dec 7, 2021

Found @nixbitcoinorg recently and it's real cool. Quite different from any other Linux distro and package manager I've ever used but damn clean.

1 2 8

nix-bitcoin Retweeted

BTC Yooper ⚡ @btcYooper · Jan 16

Tried many node setups but @nixbitcoinorg is the one that best fits my needs. Highly configurable with security by default and the maintainers are very helpful. Sent a small ⚡ to their @BtcpayServer and will continue as I keep getting value from the project.

1 1 3 342

Show this thread

nix-bitcoin Retweeted

radix rat ⛏️🦬🍁 @radixrat · Nov 3, 2021

Replying to @ODELL @nixbitcoinorg and 2 others

Setup a nix-bitcoin node after listening, its a bit more technical than a raspiblitz for OS setup IMO, but feels more minimal/locked down to only what I enabled. Definitely need mempool module though.

2 1 8

Jim Dennis

@answrguy

@nixbitcoinorg is an awesome integration of Bitcoin-core, @Liquid_BTC. @Core_LN, @lightning LND & TOR.

8:52 AM · Aug 10, 2023 · **24** Views

1

nix-bitcoin Retweeted

Ben Arc 🏴‍☠️✊⚡ @arcbtc · Jun 29, 2022

After Bitcoin Core, @nixbitcoinorg is the most important project in #bitcoin ₿.
Reproducible builds and deployments, EXACTLY what we should expect from all nodes.

youtube.com
nix-bitcoin: building a purely functional Bitcoin e...
From Understanding Bitcoin Conference, Malta, Apr 5 2019nix-bitcoin: https://github.com/fort-...

6 13 46

Why “damn clean, configurable, secure by default, minimal, reproducible”?

One crucial reason is that it’s build on NixOS!

NixOS is a Linux distribution based on the Nix package manager and build system. It supports reproducible and declarative system-wide configuration management as well as atomic upgrades and rollbacks, although it can additionally support imperative package and user management. In NixOS, all components of the distribution — including the kernel, installed packages and system configuration files — are built by Nix from pure functions called Nix expressions.

System-wide Config Management in Nix

nix
tools

configuration.nix (text files)

configuration.nix

```
{ config, pkgs, ... }: {
  imports = [
    ./hardware-configuration.nix
  ];
  services.openssh.enable = true;
  services.openssh.port = 2121;
  services.tor.hiddenServices.openssh = {
    map = [{port = config.services.openssh.port;}];
  };
}
```

```
$ nixos-rebuild switch
```

System-wide Config Management in NixOS

1. The whole system config is just a few text files which can be kept under git version control.
2. Can easily deploy a configuration to a different machine
3. Nix expressions can be checked for well-formedness

```
services.bitcoind.port = 833w;
```

ERROR at build time

nix-bitcoin configuration.nix DEMO

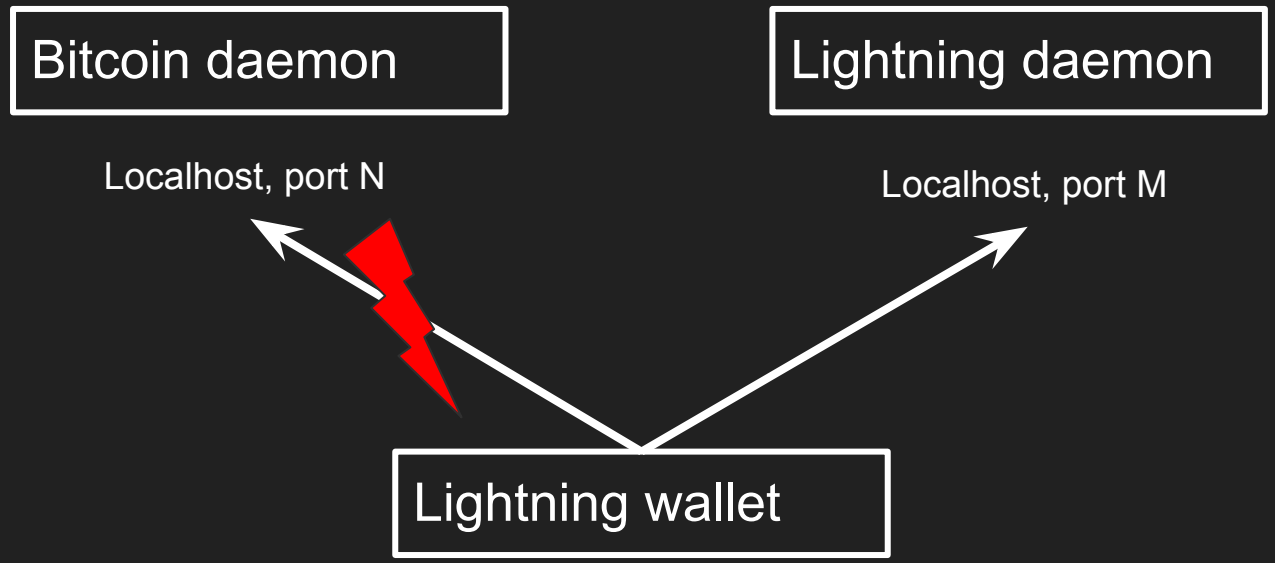
System-wide Config Management in NixOS

1. The whole system config is just a few text files which can be kept under git version control.
2. Can easily deploy a configuration to a different machine
3. Nix expressions can be checked for well-formedness

   ```
   services.bitcoind.port = 833w;
   ```
   ERROR at build time

4. Abstraction

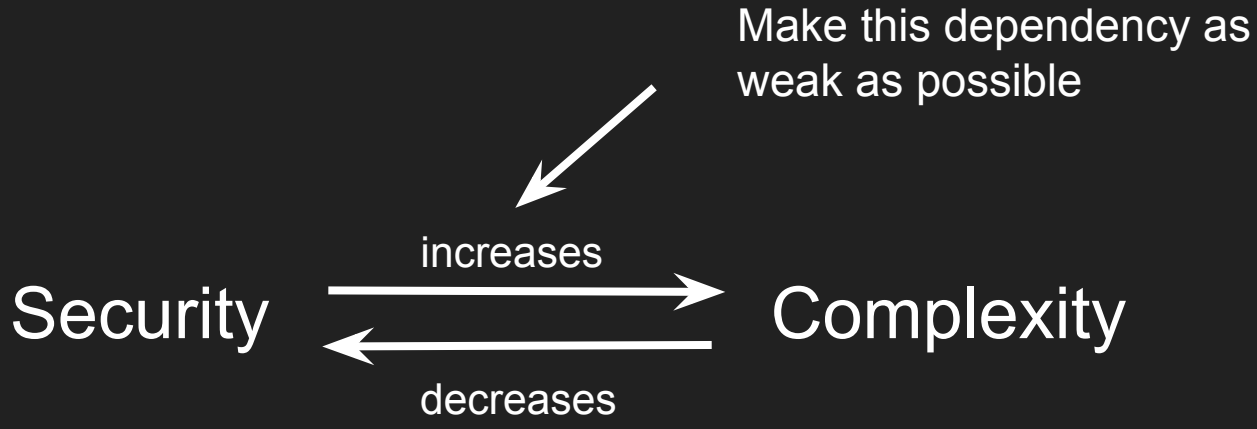
Abstraction: Network Namespace Example



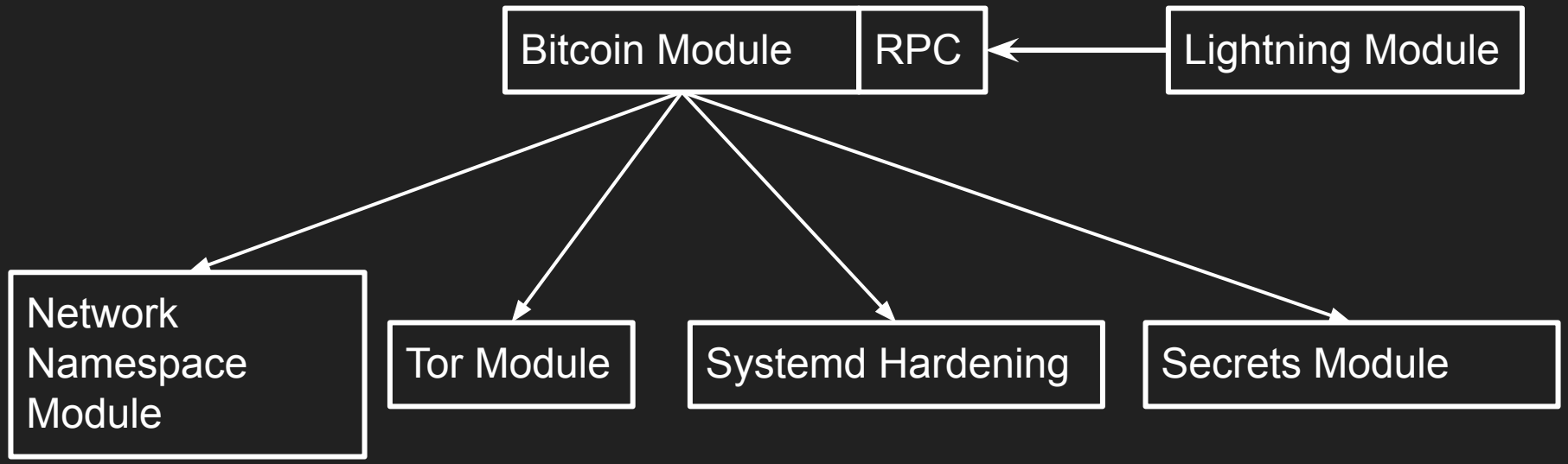
Solution: Put every service in own “network namespace” (Linux kernel feature). Then build internal router to connect namespaces.

Spaghetti code

Unstructured and hard-to-maintain code caused by lack of style rules or volatile requirements. This architecture resembles a tangled pile of spaghetti in a bowl.


Make this dependency as weak as possible
increases
Security
Complexity
decreases

Abstraction through (Nix) programming lang



Reproducibility

Reproducibility

nix
tools

Dependencies

configuration.nix (text files)

Reproducibility

Whole system config (including dependencies) is uniquely identified via hash

```
$ readlink /run/current-system

/nix/store/xqwh856sdwdlmfwqjlg17cqij5h3ljhw-nixos-system-aegon-23.11.20230927.f3dab35
```

Format: /nix/store/<hash_of_dependencies>-<name>

Reproducibility

```
$ nix-store -q --references /run/current-system

/nix/store/xj3q9xn1i32jnm8qlyckshm94gg1gzim-linux-6.1.52
/nix/store/z1gdpvkjbgv1p6d63ni26jb7fxhnfzhn-etc
<...>
```

```
/nix/store/z1gdpvkjbgv1p6d63ni26jb7fxhnfzhn-etc
-> /nix/store/0araxg0f6pmvdcs6sw8cvqv7vyqp65ww-system-units
--> /nix/store/xl73r5j1m52h47w7dsm6wad5d09h425r-unit-bitcoind.service
---> /nix/store/c2dz1bx0h1hx31a64sj8w50nlls1x0d4-bitcoind-24.1
```

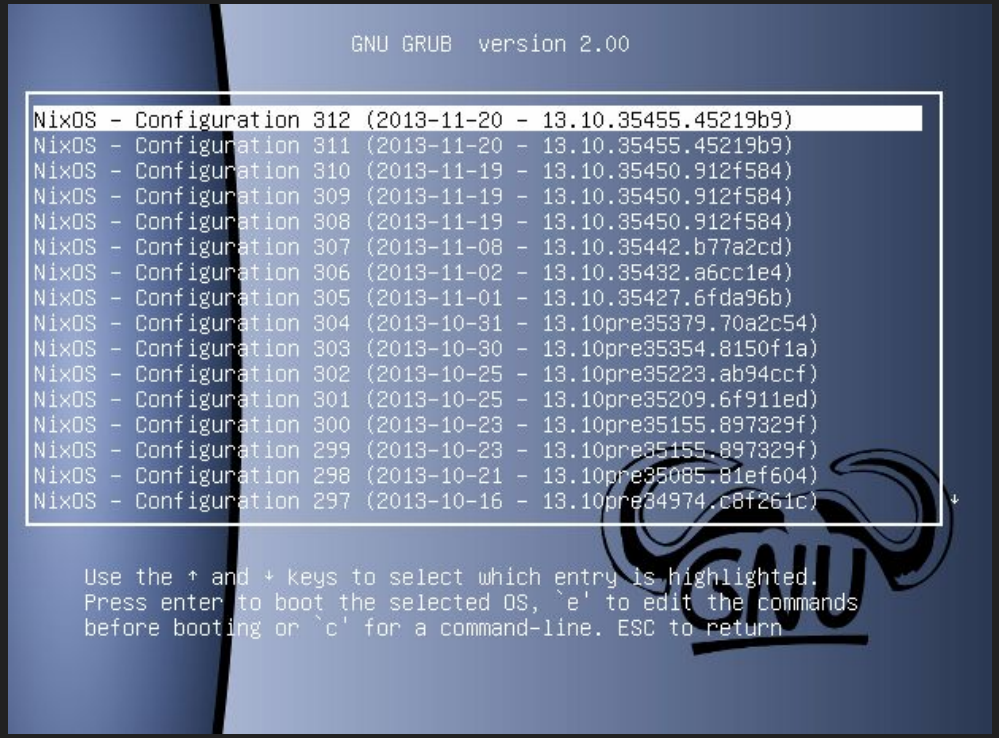
Reproducibility

```
$ nix-store -q --references …-bitcoind-24.1
/nix/store/46m4xx889wlhsdj72j38fnlyyvvvvbyb-glibc-2.37-8
/nix/store/843dqq10jdkalr2yazaz6drx334visrb-gcc-12.2.0-lib
/nix/store/5nbd10268965s5v3jwx6p2m0gh9wca57-zeromq-4.3.4
/nix/store/bzd9s7qjjwvmgcyli9cqwrfds5xx2k3r-db-4.8.30
/nix/store/mc2jjqzhpgnckrfqnmnajvni4f8hqnv1-miniupnpc-2.2.4
/nix/store/n12a68qch9s85k6ni4m4r4xxr8lwys1i-sqlite-3.41.2
/nix/store/vln6871wdfzrny7dwlrndvfxwnz78la7-libevent-2.1.12
```

Reproducibility

- We always know exactly what is running on our system
- Can build all dependencies from source to avoid trusting binary “caches”
- We won’t necessarily produce the byte-for-byte identical binaries
  - 99.77% of the NixOS minimal iso are binary reproducible.
- Updates essentially replace `/run/current-system` symlink
  - nix-bitcoin users all have the same packages, makes updates less likely to break
  - Files that are not depended on after update can be pruned

Rollbacks


GNU GRUB version 2.00
NixOS - Configuration 312 (2013-11-20 - 13.10.35455.45219b9)
NixOS - Configuration 311 (2013-11-20 - 13.10.35455.45219b9)
NixOS - Configuration 310 (2013-11-19 - 13.10.35450.912f584)
NixOS - Configuration 309 (2013-11-19 - 13.10.35450.912f584)
NixOS - Configuration 308 (2013-11-19 - 13.10.35450.912f584)
NixOS - Configuration 307 (2013-11-08 - 13.10.35442.b77a2cd)
NixOS - Configuration 306 (2013-11-02 - 13.10.35432.a6cc1e4)
NixOS - Configuration 305 (2013-11-01 - 13.10.35427.6fda96b)
NixOS - Configuration 304 (2013-10-31 - 13.10pre35379.70a2c54)
NixOS - Configuration 303 (2013-10-30 - 13.10pre35354.8150f1a)
NixOS - Configuration 302 (2013-10-25 - 13.10pre35223.ab94ccf)
NixOS - Configuration 301 (2013-10-25 - 13.10pre35209.6f911ed)
NixOS - Configuration 300 (2013-10-23 - 13.10pre35155.897329f)
NixOS - Configuration 299 (2013-10-23 - 13.10pre35155.897329f)
NixOS - Configuration 298 (2013-10-21 - 13.10pre35085.81ef604)
NixOS - Configuration 297 (2013-10-16 - 13.10pre34974.c8f261c)
Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, `e' to edit the commands
before booting or `c' for a command-line. ESC to return
GNU

nix-bitcoin Feature Highlights

- Test Framework
- Examples & nixbitcoin.org
- Hardening & Security bounty
- External modules

Test Framework

Run tests (in VM). If all tests succeed, “ready for release”.

```
$ run-tests.sh
```

Test Framework

Run test scenario:

```
$ run-tests.sh -s clightning-replication
```

Test Framework

Run command inside container

```
$ run-tests.sh -s bitcoind container --run bitcoin-cli -getinfo  # 1.5 sec
```

Test Framework

Custom test scenario (8 sec):

```
$ run-tests.sh -s "{
    services.clightning.enable = true;
    services.clightning.port = 5000;
    nix-bitcoin.onionServices.clightning.public = true;
    nix-bitcoin.onionServices.clightning.externalPort = 9735;
}" container --run c lightning-cli getinfo # Shows localhost port and external port
```

See dev/ directory for more

“Examples” & nixbitcoin.org

```
$ nix run github:fort-nix/nix-bitcoin
$ ./examples/deploy-qemu-vm.sh -i
```

Production-grade example: nixbitcoin.org on github

- Hosts webpage, custom btcpayserver-powered donation page, matrix homeserver
- Flakes-based configuration & follows best practices

Systemd Hardening

```
{
    # Default Process Config
    User = <some dedicated user>;
    ProtectSystem = "strict";                  # file system is mounted read-only
    ReadWritePaths = [ cfg.dataDir ];      # allow writes only to dataDir
    IPAddressDeny = "any";
    IPAddressAllow = [ "127.0.0.1/32" ]; # allow connections only from and to Tor
                                                    # Tor by default with fine-grained controls
    # and many more…
    NetworkNamespacePath=<...> # If netns is enabled
}
```

Bitcoind Hardening

```
# Define RPC user "public” which is allowed to only use read-only commands
services.bitcoind = [
        rpc.users.public = {
            passwordHMACFromFile = true;
            rpcwhitelist = import ./bitcoind-rpc-public-whitelist.nix;
        };
}
```

Security Bounty Fund

- Relatively well-defined and well-funded security bounty fund (2-of-3 multisig).
- Even covers certain dependencies of nix-bitcoin if they don't have their own bug bounty program.
- No payouts in the 18 months since inception.

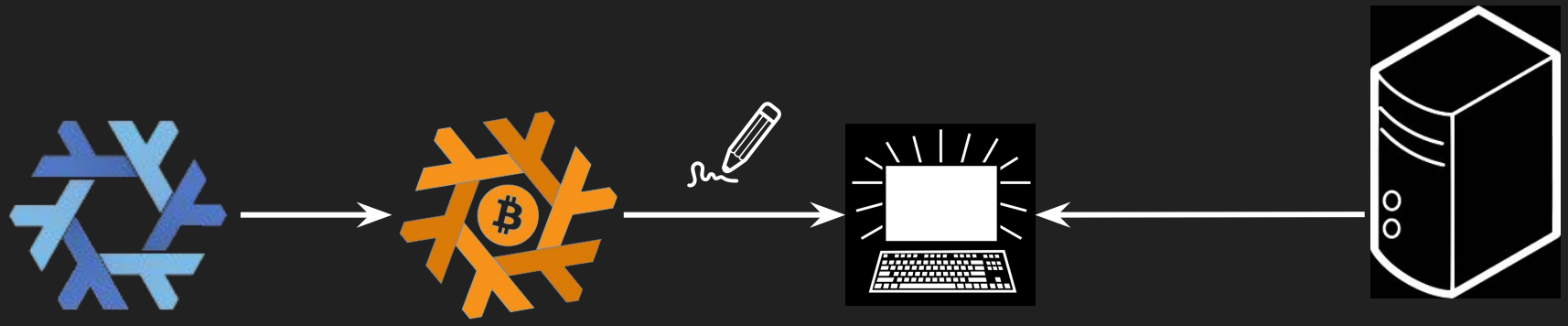
Extension Modules

It’s possible to extend nix-bitcoin with modules that are maintained outside of the nix-bitcoin repo (mempool example) and have their own review and release process.

- Removes reliance on nix-bitcoin maintainer as gatekeepers (but extension module could later evolve into proper module)
- We built extension modules such that they can use nix-bitcoin libraries and modules and they can be tested within nix-bitcoin test framework

nix-bitcoin weaknesses

nix-bitcoin weaknesses:
Insufficient Supply Chain Integrity



nixpkgs:
a giant repo with 100s (?) of people that have commit access

nix-bitcoin

node

Binary cache:
setting `substitute = false`
often breaks builds

I think these problems can be addressed

Vision!?

Keep nix-bitcoin

- **up-to-date**
- **minimal** and **low maintenance**
- **reliable**
- **extensible**

Vision!?

We’re open to adding new packages and modules for new software projects if

1. the new dependency doesn’t have a crazy build system
2. there is reasonable demand from users to add it
3. there is reason to believe that the new dependency is being maintained in the future
4. **there is a reliable volunteer who maintains the new nix-bitcoin code**

Vision!?

Nix-bitcoin is highly customizable and extensible, **let's see how it turns out to be used**!

- (Maybe someone makes user friendly (GUI?) node distribution based on nix-bitcoin)

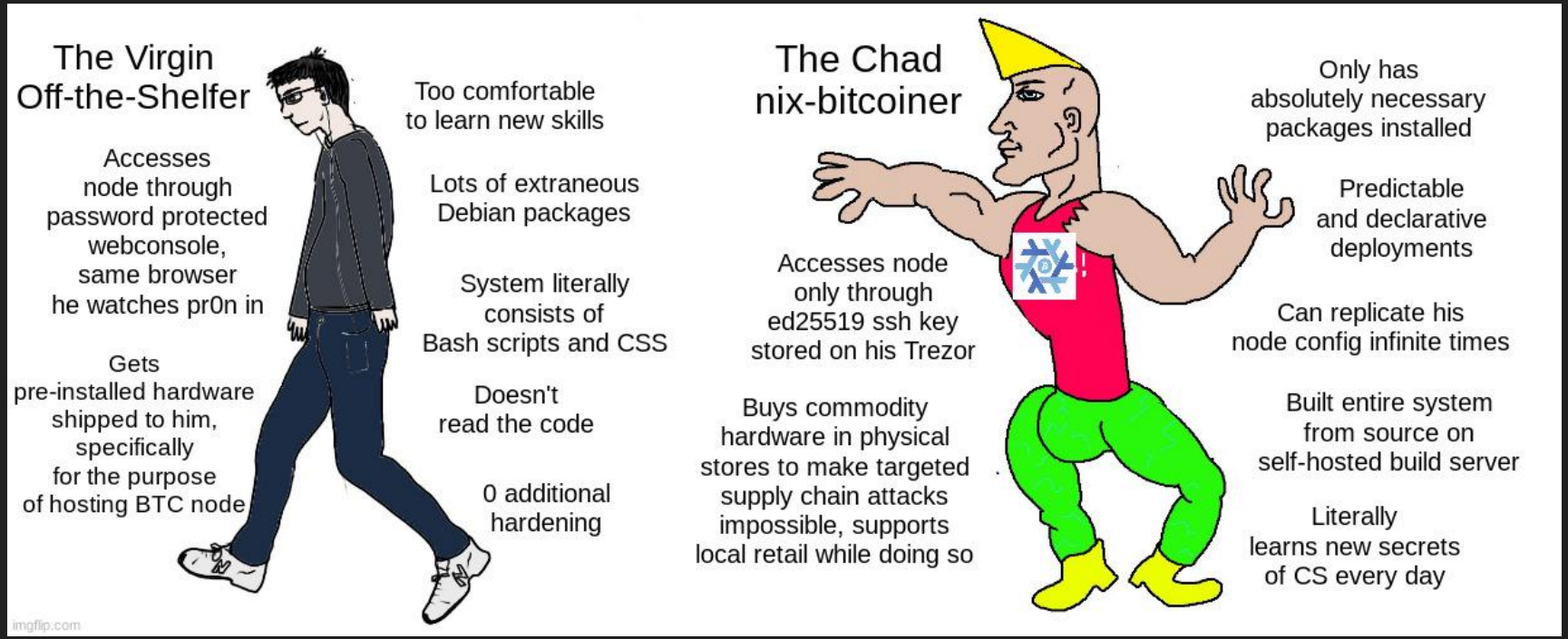
Short Term Goals

- Improve nix-bitcoin installation tutorials
- Get a proper logo!
- Add signet support
- Review mempool PR
- Add fedimint (server) module?

Before you start, some clarification

- Nix: package manager, available on most Linux distributions and MacOS
  - There’s a new feature called flakes, which basically revamped the UX of nix entirely
- NixOS: operating system, built on Nix
- Nixpkgs: “official” collection of Nix packages and NixOS modules

```
nix run github:fort-nix/nix-bitcoin
```



Slides at nickler.ninja/slides/2023-btcpp.pdf

nixbitcoin.org
Examples
Getting Started guide
Nix-bitcoin community

Zero-to-nix
NixOS manual
[nix.dev](nix.dev)
Nix pills