Network modeling with NetBox

Tom Ryder
tom@sanctum.geek.nz
https://sanctum.geek.nz/

IP Address Management (IPAM)

- Even a modest home lab now might use many IP **addresses**:
  - For a home lab, legacy IP addresses probably in an RFC 1918 subnet, like `192.168.1.0/24`, or `10.0.0.0/8`…
- Keep track of which IP addresses are on which machine
- Organise IP addresses into **prefixes**
- Both standard (IPv6) and legacy (IPv4)

Datacenter Infrastructure Management (DCIM)

- Similar software allowed you to track what's in your server room:
  - Physical hosts…
  - Rack space…
  - Cable connections…
  - Power equipment…

But wait, there’s more…

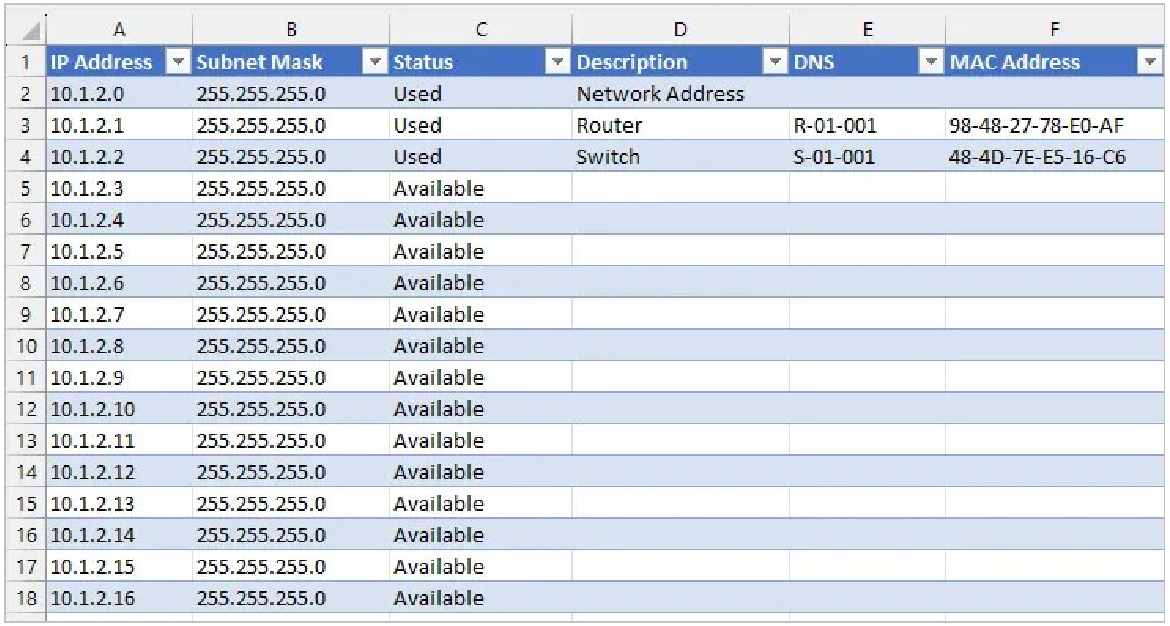
- But you might like to track many other properties of even a small home network:
  - Virtual machines…
  - VLANs (802.1Q)…
  - VPNs…
  - Tunnels…

For humans and machines

- Ideally, the system functions as a **single source of truth** about the network’s state.
  - “If it’s not in there, it’s not real yet.”
- *Humans* consult it to find inventory, spare addresses…
- *Computers* consult it in much the same way, consuming it via an **API**.

Ideally…

…we do all that *without* using **spreadsheets** in ~~Microsoft Excel~~ *LibreOffice Calc*.



	A	B	C	D	E	F
1	IP Address	Subnet Mask	Status	Description	DNS	MAC Address
2	10.1.2.0	255.255.255.0	Used	Network Address		
3	10.1.2.1	255.255.255.0	Used	Router	R-01-001	98-48-27-78-E0-AF
4	10.1.2.2	255.255.255.0	Used	Switch	S-01-001	48-4D-7E-E5-16-C6
5	10.1.2.3	255.255.255.0	Available			
6	10.1.2.4	255.255.255.0	Available			
7	10.1.2.5	255.255.255.0	Available			
8	10.1.2.6	255.255.255.0	Available			
9	10.1.2.7	255.255.255.0	Available			
10	10.1.2.8	255.255.255.0	Available			
11	10.1.2.9	255.255.255.0	Available			
12	10.1.2.10	255.255.255.0	Available			
13	10.1.2.11	255.255.255.0	Available			
14	10.1.2.12	255.255.255.0	Available			
15	10.1.2.13	255.255.255.0	Available			
16	10.1.2.14	255.255.255.0	Available			
17	10.1.2.15	255.255.255.0	Available			
18	10.1.2.16	255.255.255.0	Available			

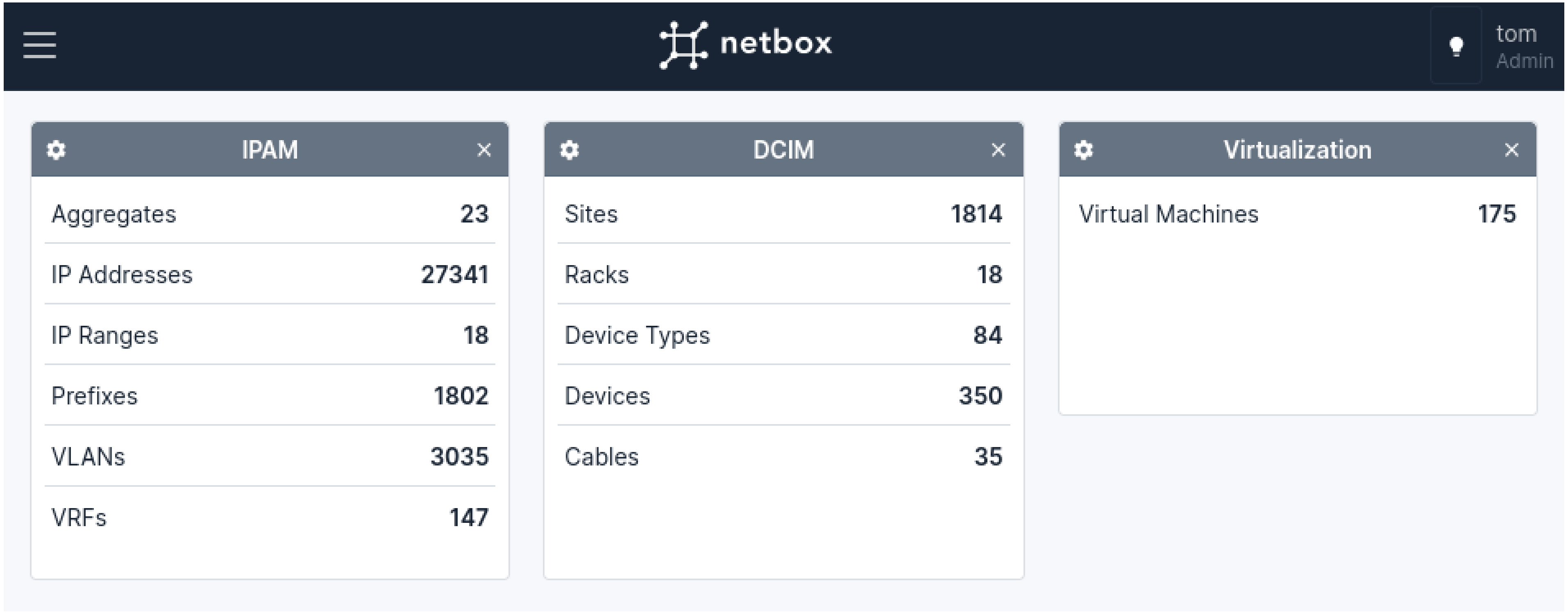
Enter NetBox—1/2

- “The Premier Network Source of Truth”, for “modeling and documenting modern networks.”
- Very many object types
  - Most extensible with custom fields
- REST API for objects (read-write)
- Regularly updated
- Python (Django-based) web application

Enter NetBox—2/2

- There’s an “Enterprise” version they host for you with support…
- …but it is **free and open-source software**, when installed self-hosted.
  - License is Apache-2.0 (permissive)

Scales nicely



IPAM	
Aggregates	23
IP Addresses	27341
IP Ranges	18
Prefixes	1802
VLANs	3035
VRFs	147

DCIM	
Sites	1814
Racks	18
Device Types	84
Devices	350
Cables	35

Virtualization	
Virtual Machines	175

Public demo

- There’s a public demo of NetBox at https://demo.netbox.dev/
- Create an account, log in, and try it out without having to install.
- It gets reset every day at 04:00 UTC, so you can’t really break anything.

Supported platforms

- **Ubuntu Linux** and **CentOS** are supported, and have installation instructions.
- However, I had no problems making it work on my **Debian GNU/Linux** server.
- NetBox should be installable on any GNU/Linux distribution with a decent set of packages…or a dedicated administrator.

Installation—1/2

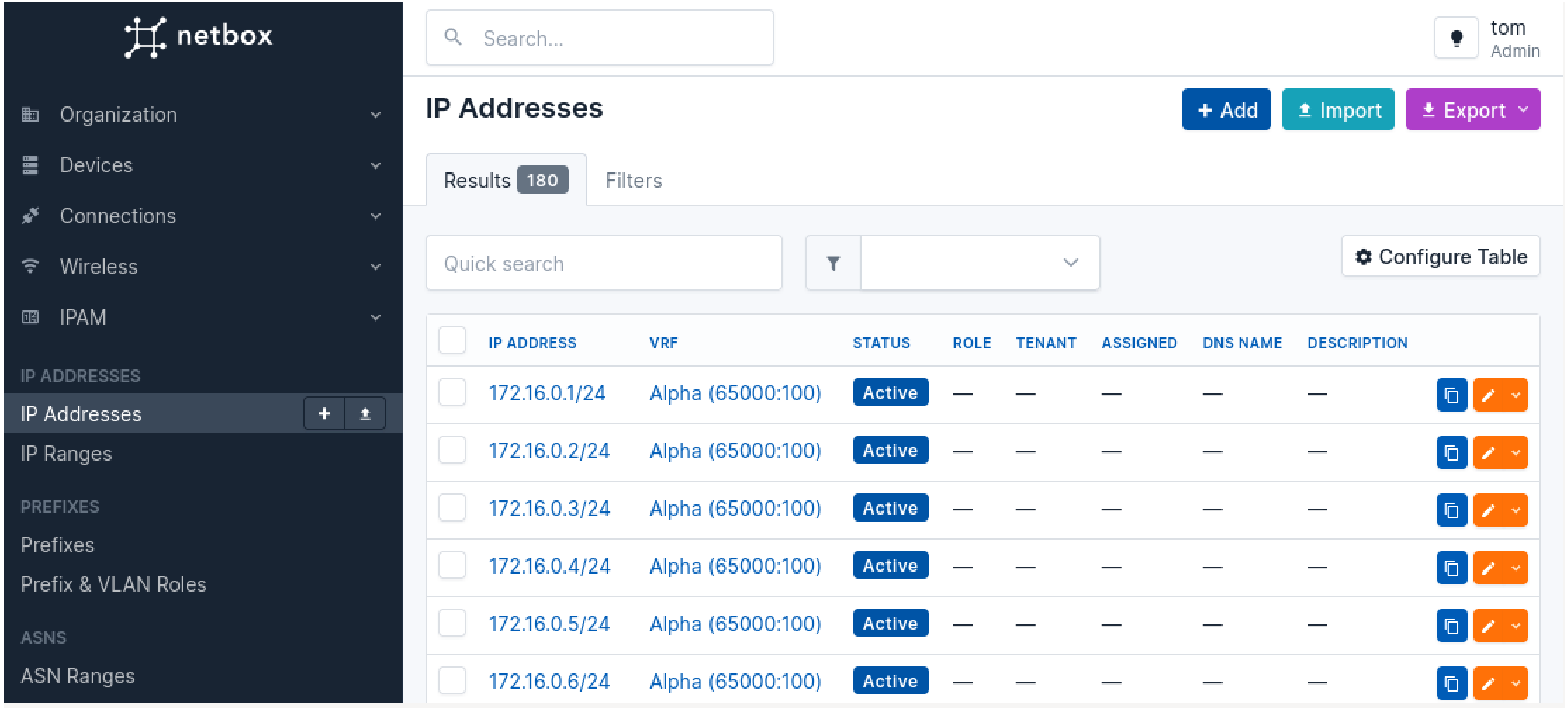
- Not exactly point-and-click, but not too bad.
  - Install **PostgreSQL** and set up database
  - Install **Redis object cache** (warning!)
  - Install **NetBox components**
  - Install **web server** (Gunicorn or uWSGI)
  - Install **reverse proxy** (Apache HTTPD or Nginx)

Aside: Redis is dead to us

- Versions of Redis v7.4 or newer are not free software.
- A free software replacement like Valkey will hopefully be better-packaged in e.g. Debian GNU/Linux very soon.
- Use that for your NetBox instance instead.
- **Don’t** depend on non-free software for your network modeling!

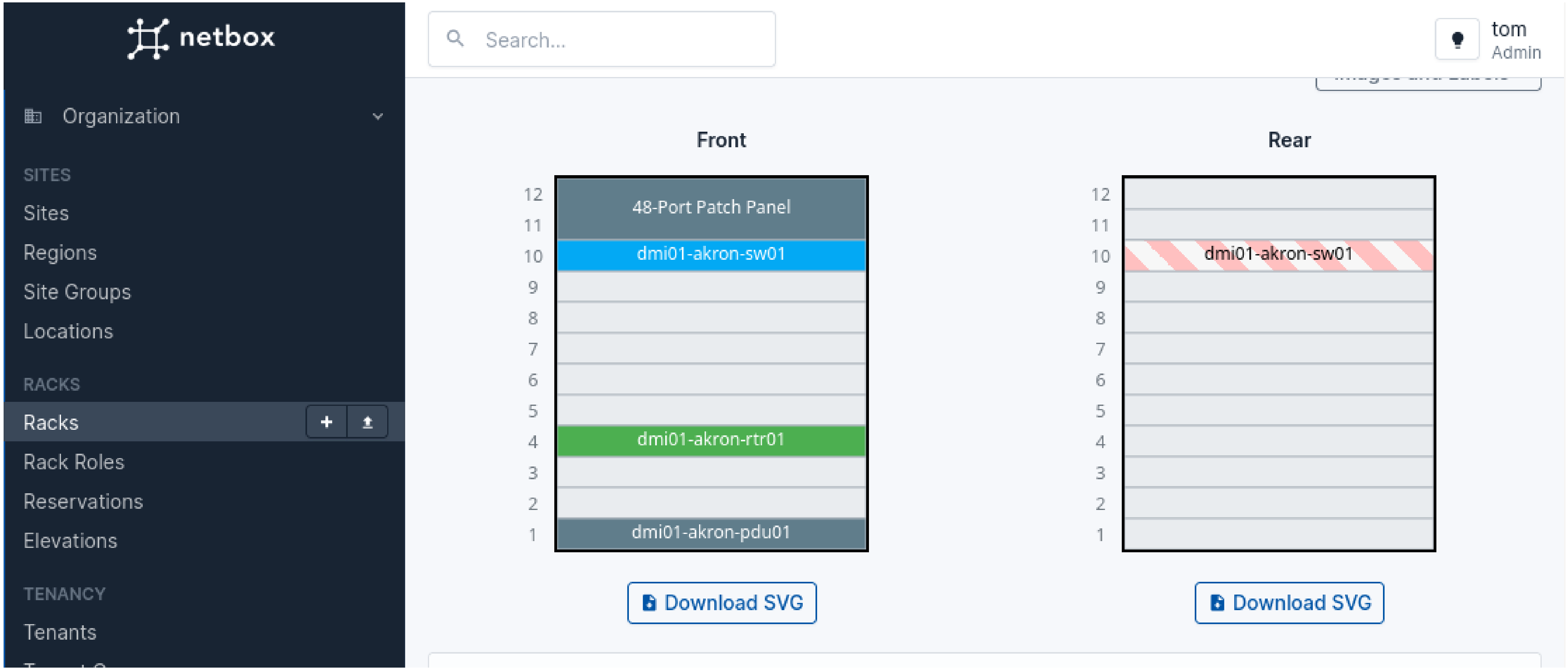
Installation—2/2

- There’s also a Docker image, if you’re into that.
- Tom is *not* into that.™



IP Addresses

	IP ADDRESS	VRF	STATUS	ROLE	TENANT	ASSIGNED	DNS NAME	DESCRIPTION
	172.16.0.1/24	Alpha (65000:100)	Active	—	—	—	—	—
	172.16.0.2/24	Alpha (65000:100)	Active	—	—	—	—	—
	172.16.0.3/24	Alpha (65000:100)	Active	—	—	—	—	—
	172.16.0.4/24	Alpha (65000:100)	Active	—	—	—	—	—
	172.16.0.5/24	Alpha (65000:100)	Active	—	—	—	—	—
	172.16.0.6/24	Alpha (65000:100)	Active	—	—	—	—	—


netbox
Search...
tom
Admin
Organization
SITES
Sites
Regions
Site Groups
Locations
RACKS
Racks
Rack Roles
Reservations
Elevations
TENANCY
Tenants
Front
Rear
12
11
10
9
8
7
6
5
4
3
2
1
48-Port Patch Panel
dmi01-akron-sw01
dmi01-akron-rtr01
dmi01-akron-pdu01
dmi01-akron-sw01
Download SVG
Download SVG

Demo

- We’ll log in to, and click around in, an instance of NetBox v4.0.8.
- It’s loaded it up with demo data so we can get a better idea of how it all looks.
- This will be audience-directed: **call out** if you want Tom to click on anything that looks interesting.

REST API—1/3

- *Don’t* interact directly with the PostgreSQL database, especially if you’re POSTing or PUTting.
- Use a **REST API** (HTTP requests) to create, update, and delete any record type.
- All done with **JSON**, in Django style.

REST API—2/3

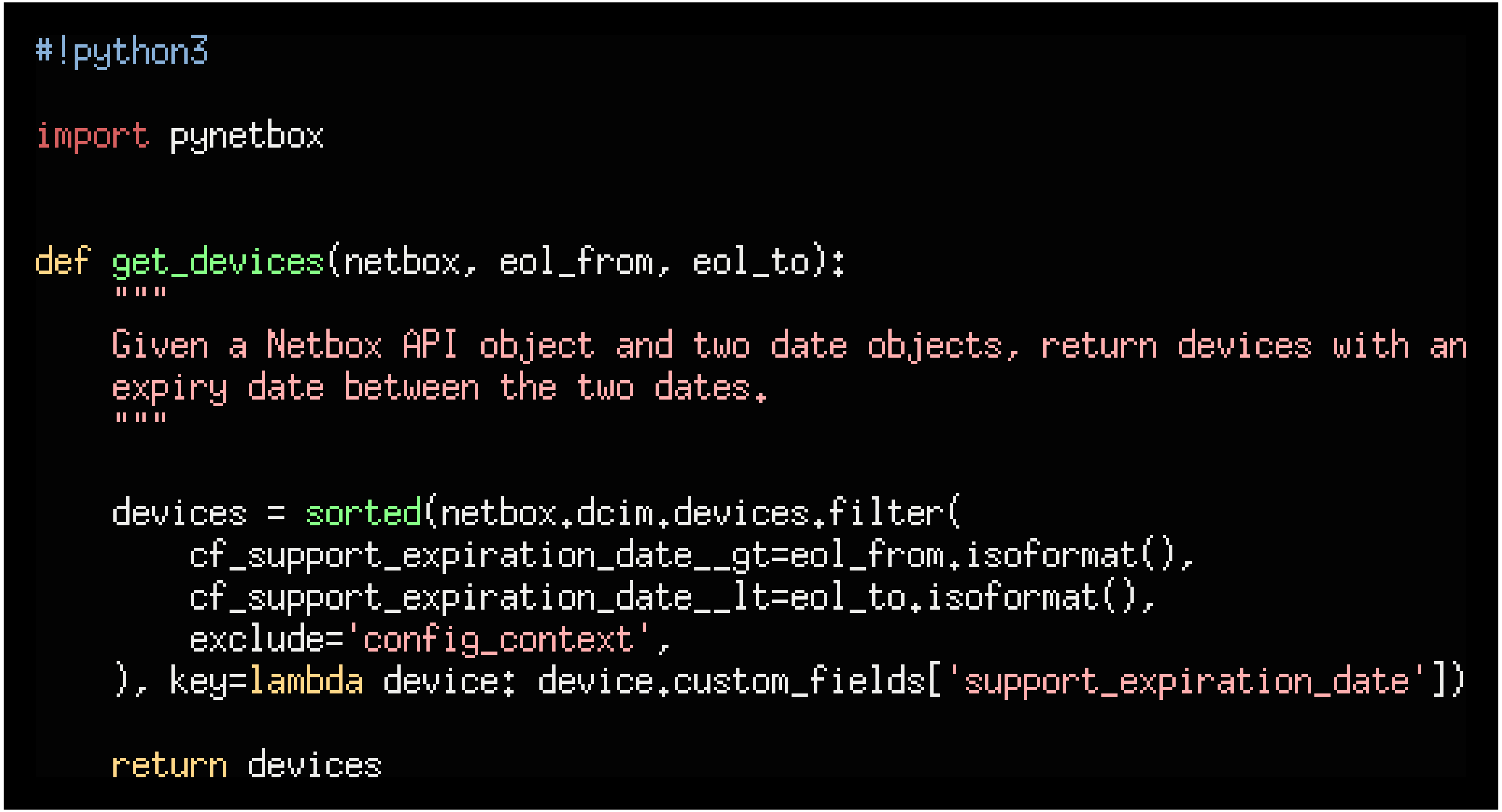
- You can just use cURL for ad-hoc, one-off requests; it works fine:

```
$ curl -H 'Authorization: Token
123abc…' -H 'Accept:
application/json'
https://netbox.example.com/api/ipam/v
lans/ | jq
```

REST API—3/3

- However, it's probably best to use a library.
- If there's a dedicated NetBox module for your language—great, use that!
  - pynetbox has served me well so far:

    ```
    $ pip install pynetbox
    ```
- But any general HTTP REST client library that supports JSON should do just fine. There aren't any nasty surprises.



```python
#!python3

import pynetbox


def get_devices(netbox, eol_from, eol_to):
    """
    Given a Netbox API object and two date objects, return devices with an
    expiry date between the two dates.
    """

    devices = sorted(netbox.dcim.devices.filter(
        cf_support_expiration_date__gt=eol_from.isoformat(),
        cf_support_expiration_date__lt=eol_to.isoformat(),
        exclude='config_context',
    ), key=lambda device: device.custom_fields['support_expiration_date'])

    return devices
```

Ideas

- Generate **monitoring system config** (Nagios Core, Icinga…)
- Generate **DNS zones**
- Use as the basis for **customer billing**
- Use with **config management** (Ansible, Chef, Puppet…)
  - Look up what this server is supposed to do in NetBox?
  - Sure, why not?

Questions?

- Docs
- Demo data
- Python module
- Single source of truth
- Awesome NetBox

Email: tom@sanctum.geek.nz
Website: https://sanctum.geek.nz/
Fediverse: @tejr@mastodon.sdf.org