F-Root on BSD: Alpha to Now

Tales of engineering in production

Dan Mahoney/BSDCan 2025 (dmahoney@isc.org)

Requisite Slide: XKCD 705

WE TOOK THE HOSTAGES, SECURED THE BUILDING, AND CUT THE COMMUNICATION LINES LIKE YOU SAID.
EXCELLENT.
BUT THEN THIS GUY CLIMBED UP THE VENTILATION DUCTS AND WALKED ACROSS BROKEN GLASS, KILLING ANYONE WE SENT TO STOP HIM.
AND HE RESCUED THE HOSTAGES?
NO, HE IGNORED THEM. HE JUST RECONNECTED THE CABLES WE CUT, MUTTERING SOMETHING ABOUT "UPTIME".
SHIT, WE'RE DEALING WITH A *SYSADMIN*.

Quick Housekeeping things

- For questions, if someone else can watch for hands, I would appreciate it.
- If not, just find a place to interrupt.
- My sunglasses are prescription, so if you see me dropping them, it’s just so I can see you, not to hide my side-eye.
- This talk is mostly about FreeBSD, but if you have questions about routing, peering, BGP, catch me afterward?

About Me

- Started at a Dialup ISP in 1999
- Dedicated server farm for a few years, mainly systems-focused. Our ticket system had a “This is a Dan Problem” button in it.
- Started with ISC in 2008, continued through several re-orgs.
    - Maintaining internal systems
    - Care and Feeding of our VMWare infra
    - Mail operations
    - And of course, F-root

About my relationship with BSD

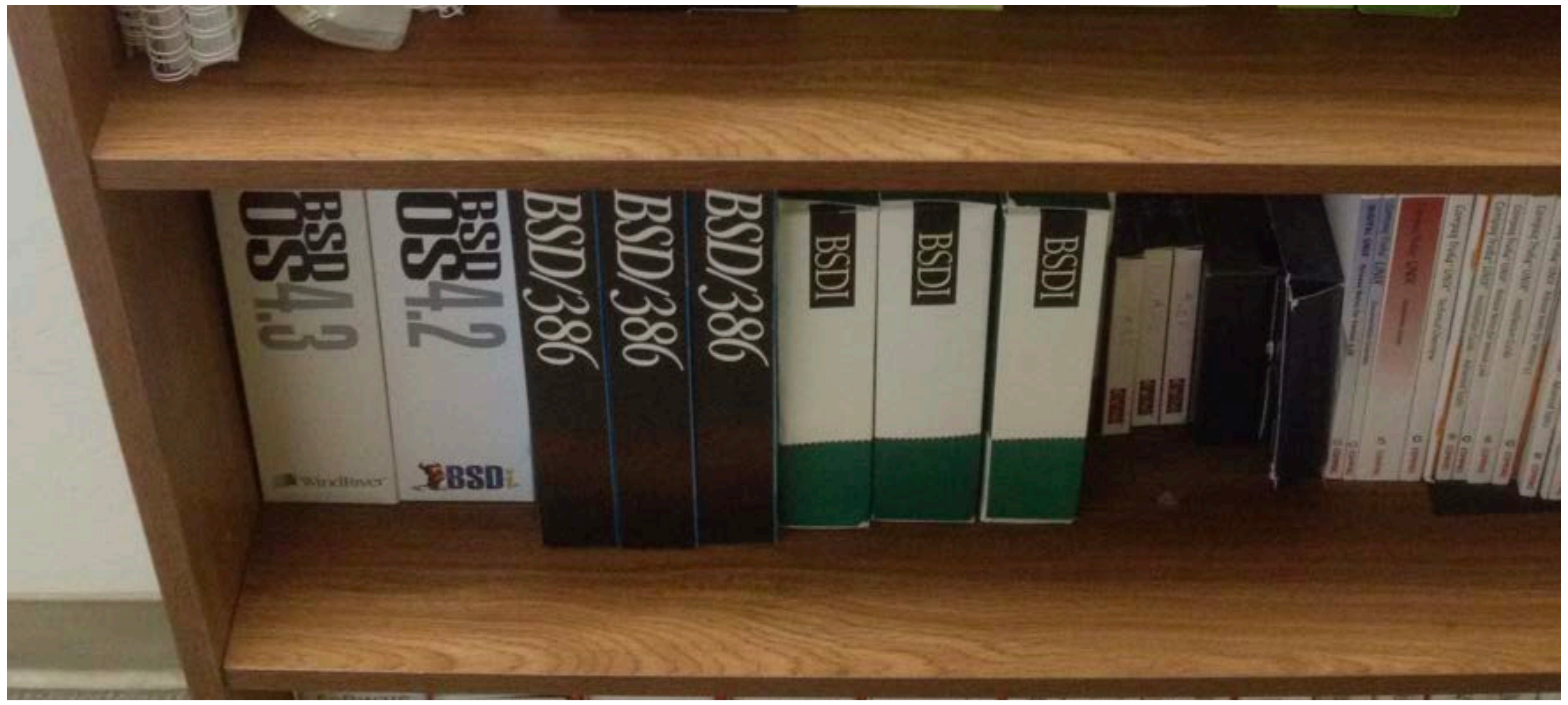
This is not a linux-bashing slide

- There’s a whole “Linux Documentation Project” website (tldp.org). The FreeBSD documentation project is called FreeBSD.org
- Lack of a cohesive user community in Linux.
- Often found that the answer to Linux questions was “Why do you want to do that, that’s dumb!” or “well, here’s how you do it on this other distro.”
- Actual man pages, sane interface names, sane startup scripting, sane network config, not stories about how things like “ifconfig” are deprecated but still installed, and still work (and are not just aliases for the new tools, or a wrapper that print new-tool usage).
- BSD just let me get what I needed to do, done.
- Two other members of my team seem to have embraced the elegance of the operating system, both running it now for their own purposes.

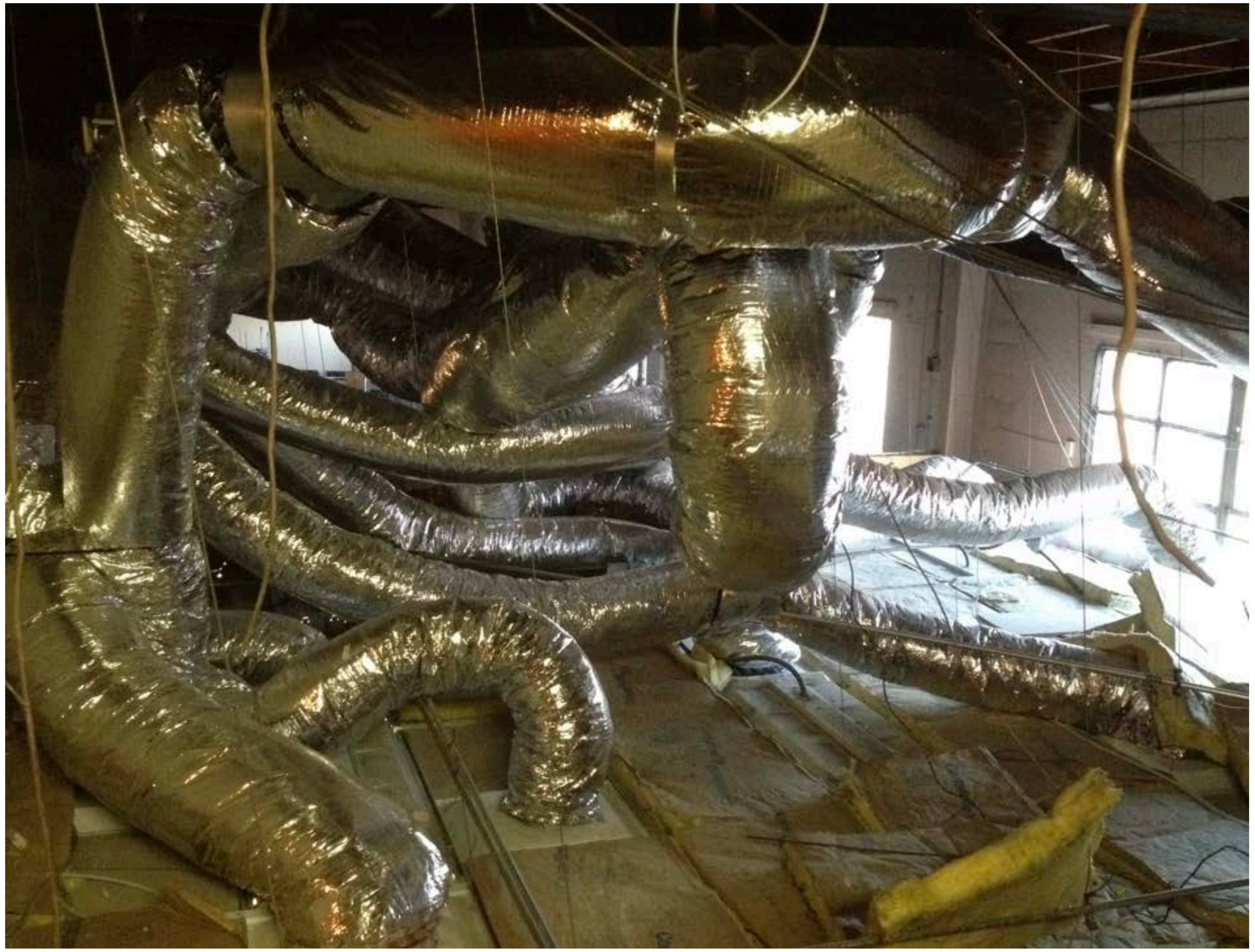
About ISC (and FreeBSD)

- We’re a 501(c)(3) non-profit formerly based in Redwood City, California. (Now, fully virtual, with corporate folks in NH). A lot of our early people came from DEC or PAIX.
- We make BIND, and Two DHCP servers, and give it away for free, but charge for support.
- And we run a root DNS server (on BSD since inception), where the user base is the entire DNS-using world.
- There’s a long history and it’s not all on our website, but we’ve been known to the FreeBSD project for a long time, have often hosted mirrors and project infra.
- I’m not the person to tell all the stories, but if you ever meet Paul Vixie, buy him a beer and ask.

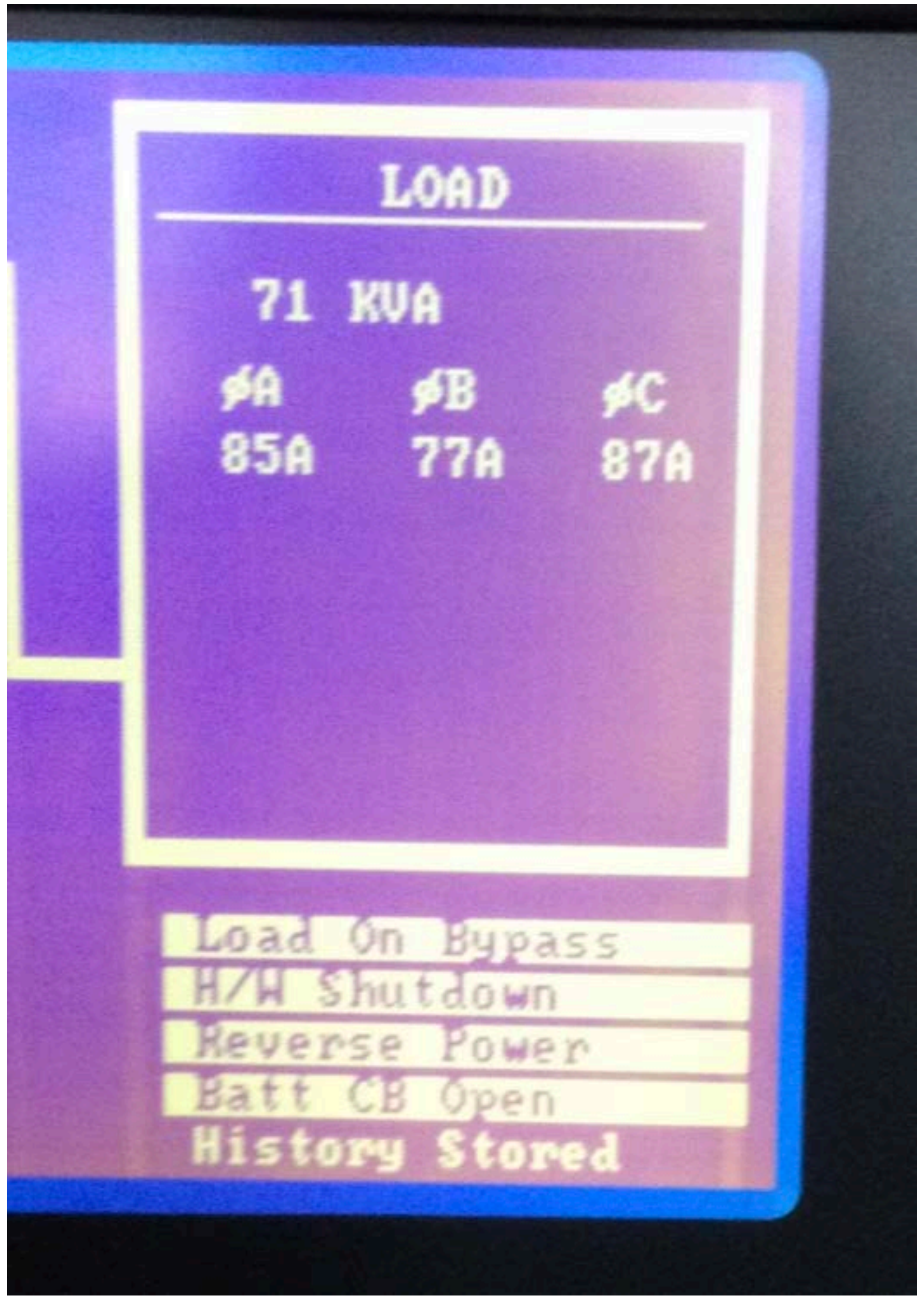
A bookshelf at our old HQ



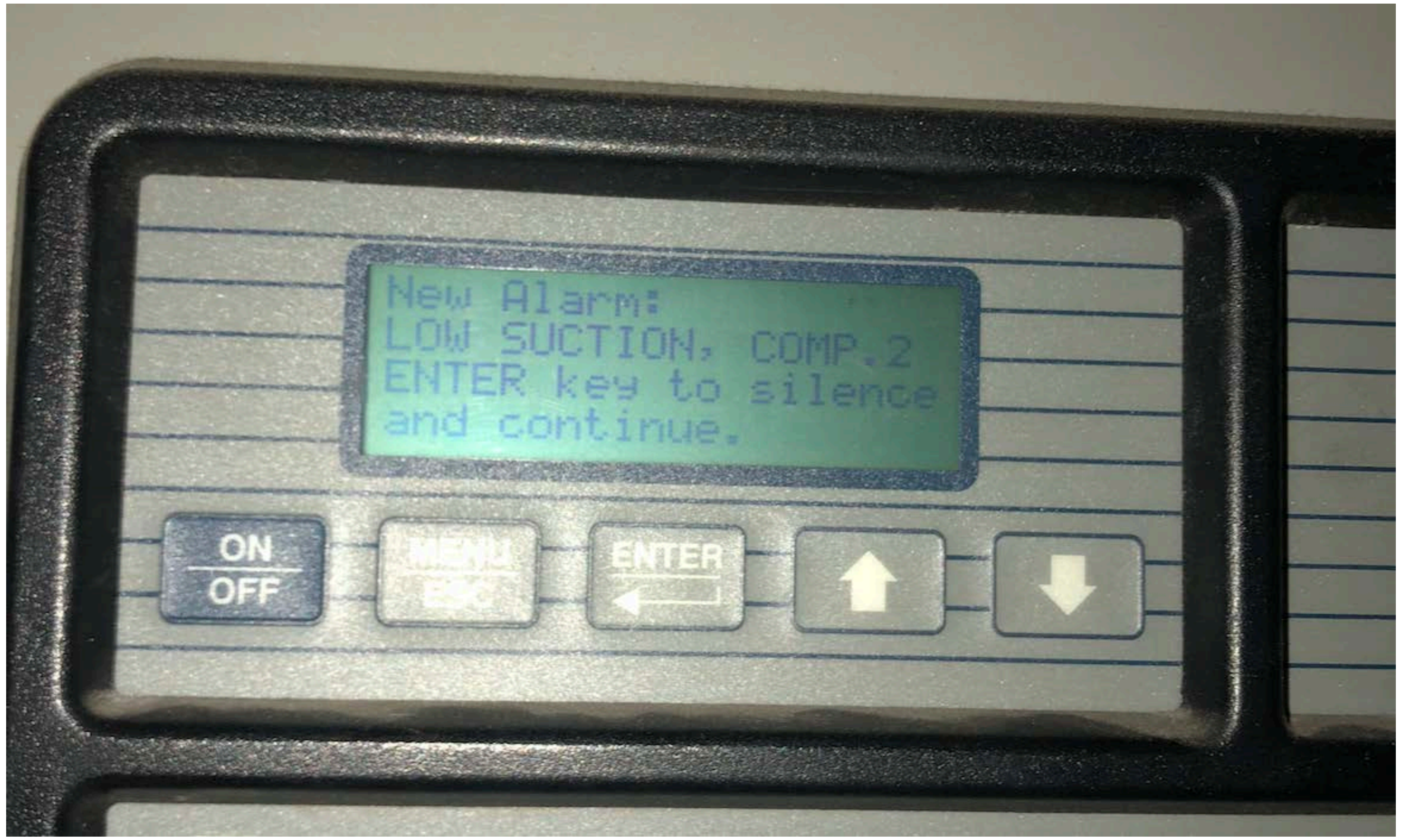
Brief Aside: 950 Charter Street



The Building did not like to be left alone

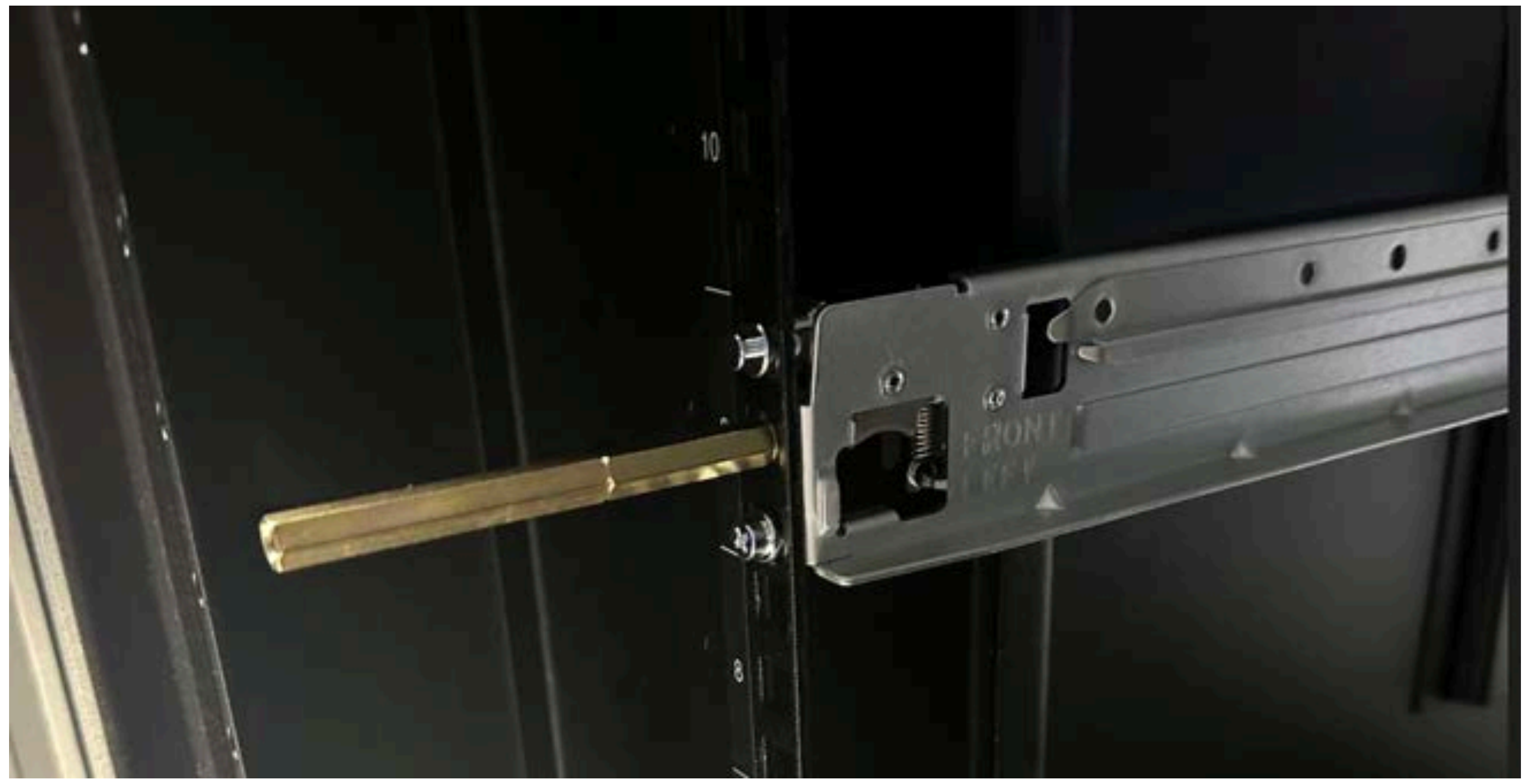


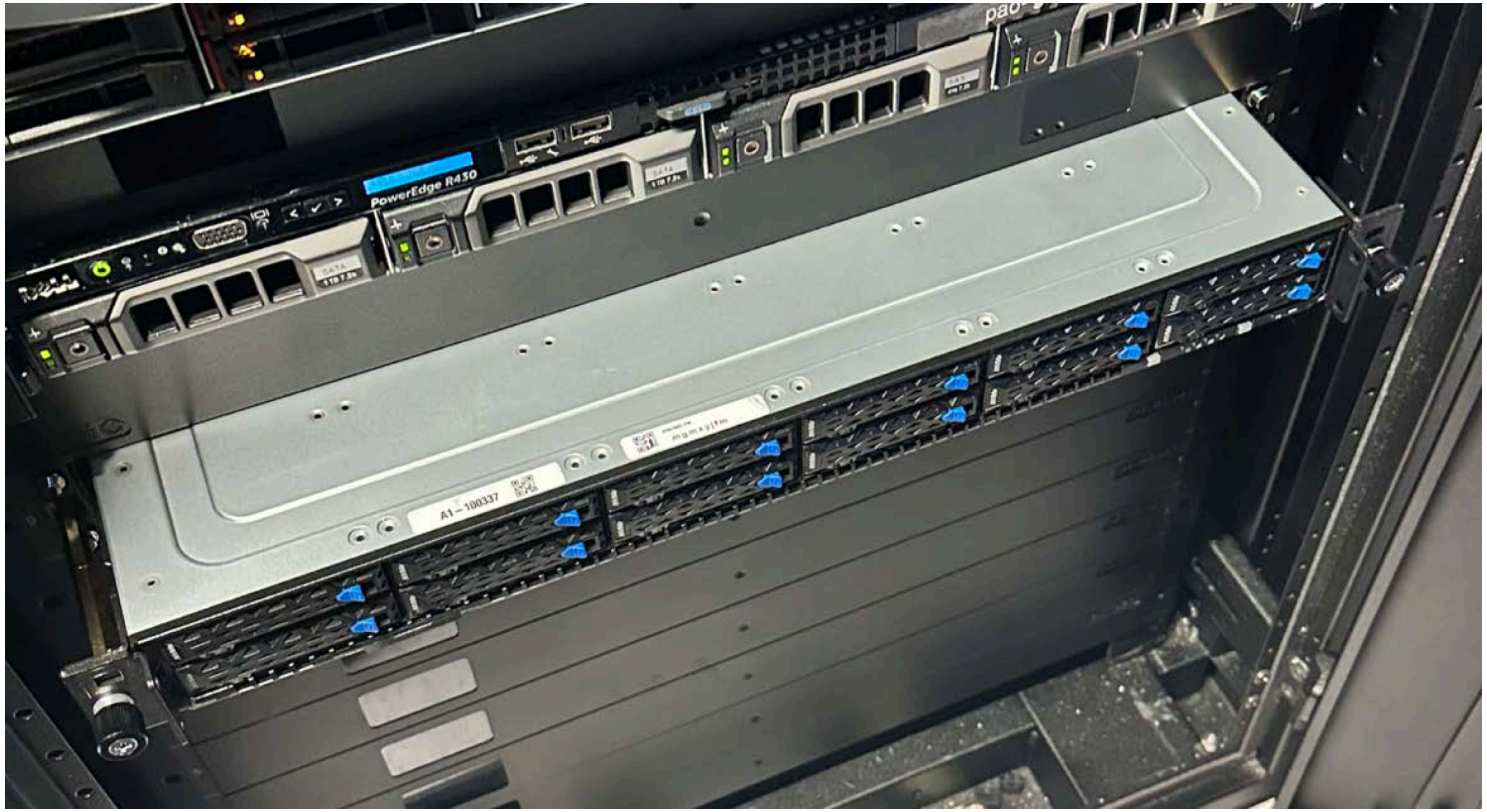
```
LOAD

71 KVA
ϕA     ϕB     ϕC
85A    77A    87A

Load On Bypass
H/W Shutdown
Reverse Power
Batt CB Open
History Stored
```



```
New Alarm:
LOW SUCTION, COMP.2
ENTER key to silence
and continue.
```

Speaking of Which





What a root DNS server is

- Root DNS servers are responsible for the apex of the DNS.
- When you send a raw query to an unprimed resolver, it has no idea where to find mail.google.com, and your query goes to the root, who says "I don't know, but .com is handled by these guys, their addresses are x.x.x.x and y:y:::yy"
- It also means we get all the queries for garbage domains like .local and .netgear and every misconfigured server and router, and means we get a lot of info about who has their network configured wrong.

What a Root DNS server used to be (long ago, before my time)

- A pair of DEC alphas, in a Rack at 529 Bryant Street (PAIX), running BSD and BIND.
- Two machines so that you could reboot one.
- Anycast came along soon, which allowed each machine to announce its availability to serve the root DNS zone into local routers using OSPF, which would be redistributed into BGP. (Vixie/Abley)
- F was the first root DNS server to do anycast.
- Global anycast followed, where F was announced from multiple locations, including small “local” nodes that were only at an IX, using the NO_EXPORT BGP Community to keep local servers reachable only by a local audience.
- Way more routing details including subnet size, CDNs, and ECMP that I won’t get into here.

What an F-Root DNS server looked like (more recent past)

- Two small Cisco/Juniper Routers (7200 or m5/m7i)
- Two Managed Switches (for redundancy)
- Three F-Root Servers (two service nodes [A/B] plus a management box [Z] to run serial consoles, typically via an off-the-shelf usb-to-8-port-de9-box)
- Most of these were deployed and managed on site by having a sysadmin visit by one of my predecessors.
- As a fallback, sometimes local hands (in different time zones, who are not used to FreeBSD), could be stepped through doing a clean OS install.
- (*foreshadowing*)

What a Root DNS Server looks like now

- As Cisco routers in the field reached end of life, and started approaching memory limitations as the global routing table grew, the state of software routing daemons like BIRD reached maturity.
- A single machine (very typically a Dell PowerEdge) running FreeBSD, with a connection for its iDRAC, a “normal” FreeBSD connection, and one or more connections to local exchanges.
- iDRAC has replaced the need for serial consoles, as well as giving us things like crash screen capture, hardware watchdogs, and the ability to set bios knobs directly. The iDRAC also exposes SNMP OID’s that show hardware health and RAID health, gives us centralized firmware updates. Buy me a rootbeer and ask me about iDRACs.
- Software routing daemons on the box have replaced the need for hardware routers.
- A second FIB lets us keep our simple, static management functions distinct from complex routes learned from BGP peers.
- Automation with Puppet.

Upgrading our systems from the old model to the new one

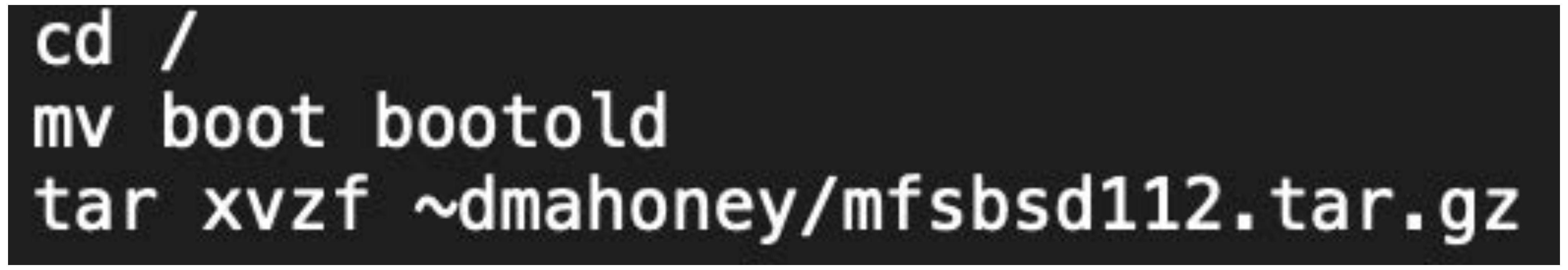
- Not a drop-in upgrade — what used to be two transit connections speaking BGP needed to be converted to management and idrac.
- We attempted to purchase Dell Machines locally and ship them out around the world.
- Managing global logistics (and VAT) is a nightmare, and every country is different. (I have stories)
- Turns out, Dell's warranty doesn't apply to boxes that have been moved out of country.
- And most of our boxes are running for free, to benefit an IX and the local community.
- So our answer: "You buy the box in-country, pass us creds, we'll operate it, the DNS wins".
- We also work with a CDN, but not as a handoff of our responsiblity — we firmly believe we should be able to stand on our own.

Sysadmin Combat Story #1: Upgrades

- The year was 2020. ISC had gone full-virtual just a few months ago.
- Nobody was buying or shipping hardware, staffing was minimal.
- Several older OS versions were going EOL, but we couldn’t get people to the datacenters to do OS reinstalls.
- Older systems with only a serial console (no boot from sdcard, no PXEboot, no BIOS console redirection, no serial console command to effect a hard reset).
- Some of them even i386.
- …and a / partition too small to hold a modern kernel (did you know that freebsd-update won’t check available disk? I do. Ask me how I know.)

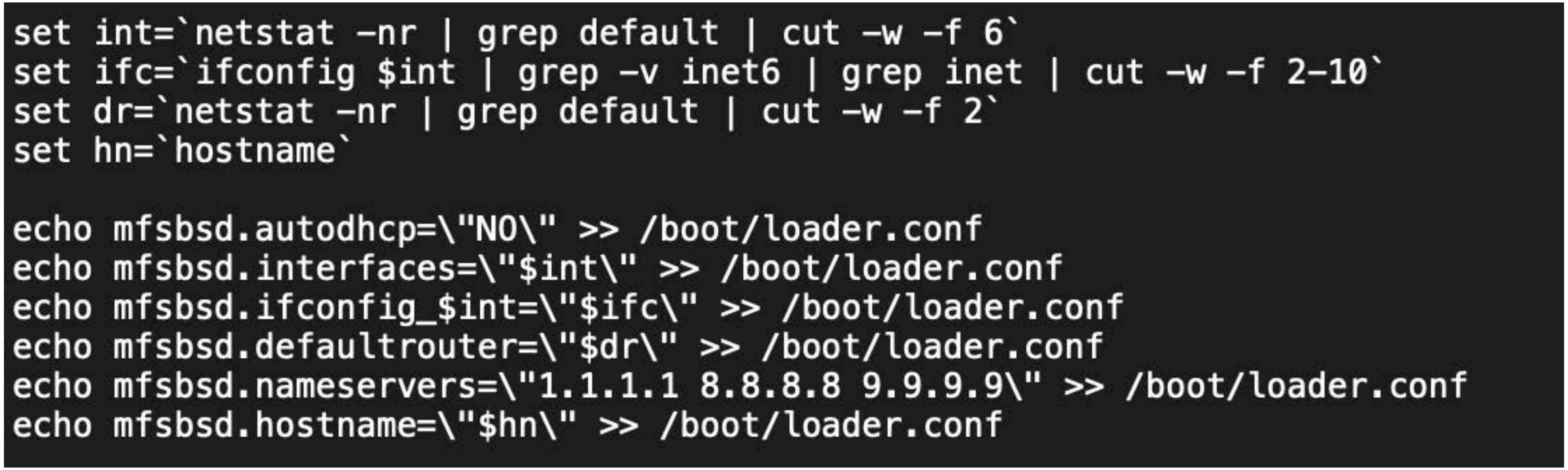
Upgrades in the time of lockdown

- Physical hardware, in the field, with an EOL OS, spotty OOB access, and your remote hands literally isn't allowed to leave their home, and if they were, there would be both a language and technology barrier.
- You have a thing that's up and working, and you are not paying them. You're not a priority.
- I tested this in a VM first but…
- Don't try this at home, kids.



```
cd /
mv boot bootold
tar xvzf ~dmahoney/mfsbsd112.tar.gz
```

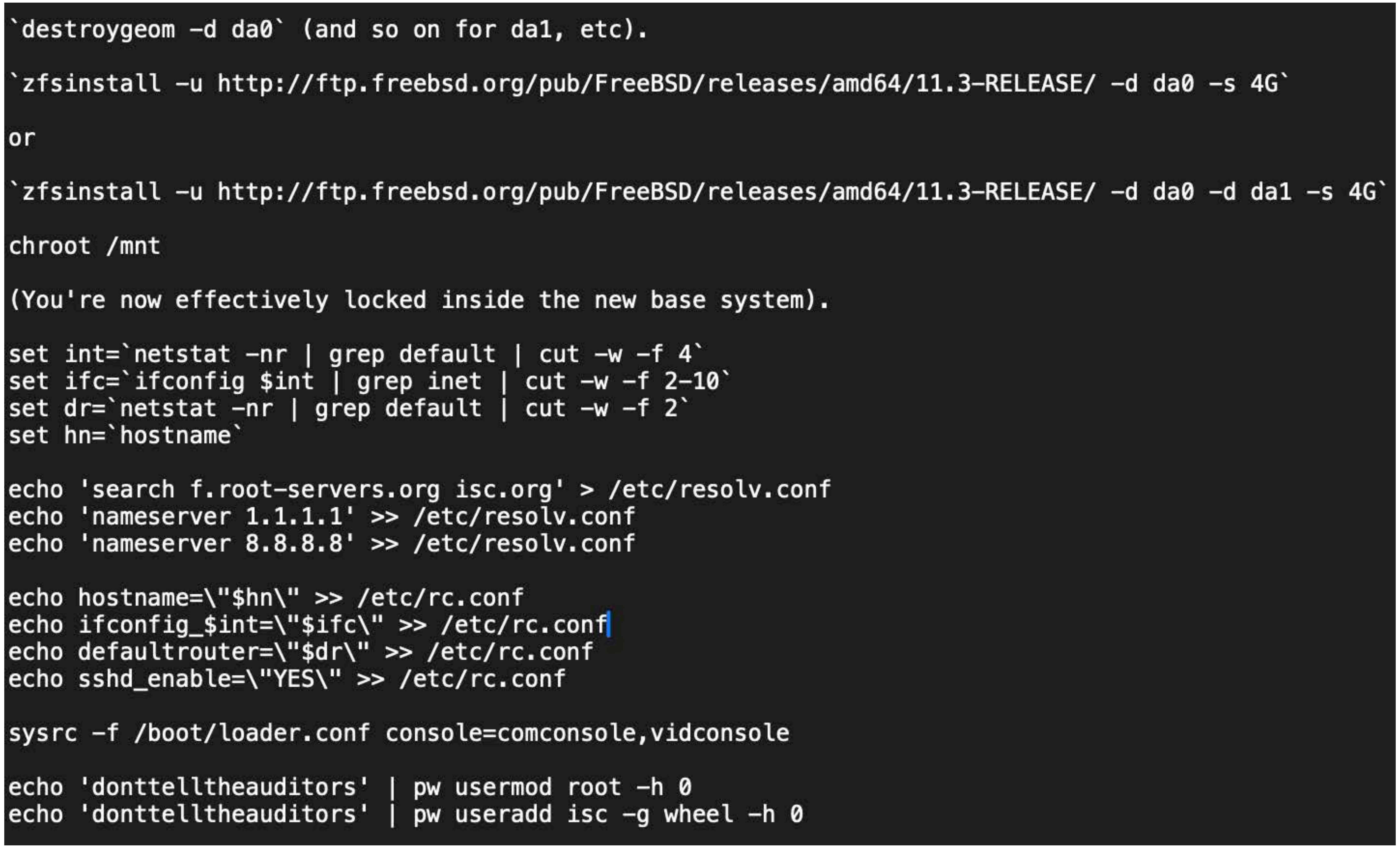
Combat Story #1: MFSBSD Lunacy



```
set int=`netstat -nr | grep default | cut -w -f 6`
set ifc=`ifconfig $int | grep -v inet6 | grep inet | cut -w -f 2-10`
set dr=`netstat -nr | grep default | cut -w -f 2`
set hn=`hostname`

echo mfsbsd.autodhcp=\"NO\" >> /boot/loader.conf
echo mfsbsd.interfaces=\"$int\" >> /boot/loader.conf
echo mfsbsd.ifconfig_$int=\"$ifc\" >> /boot/loader.conf
echo mfsbsd.defaultrouter=\"$dr\" >> /boot/loader.conf
echo mfsbsd.nameservers=\"1.1.1.1 8.8.8.8 9.9.9.9\" >> /boot/loader.conf
echo mfsbsd.hostname=\"$hn\" >> /boot/loader.conf
```

Reboot and keep a ping running and pray it comes back…

Combat Story #1: Post-Reboot MFSBSD



```
`destroygeom -d da0` (and so on for da1, etc).

`zfsinstall -u http://ftp.freebsd.org/pub/FreeBSD/releases/amd64/11.3-RELEASE/ -d da0 -s 4G`

or

`zfsinstall -u http://ftp.freebsd.org/pub/FreeBSD/releases/amd64/11.3-RELEASE/ -d da0 -d da1 -s 4G`

chroot /mnt

(You're now effectively locked inside the new base system).

set int=`netstat -nr | grep default | cut -w -f 4`
set ifc=`ifconfig $int | grep inet | cut -w -f 2-10`
set dr=`netstat -nr | grep default | cut -w -f 2`
set hn=`hostname`

echo 'search f.root-servers.org isc.org' > /etc/resolv.conf
echo 'nameserver 1.1.1.1' >> /etc/resolv.conf
echo 'nameserver 8.8.8.8' >> /etc/resolv.conf

echo hostname=\"$hn\" >> /etc/rc.conf
echo ifconfig_$int=\"$ifc\" >> /etc/rc.conf
echo defaultrouter=\"$dr\" >> /etc/rc.conf
echo sshd_enable=\"YES\" >> /etc/rc.conf

sysrc -f /boot/loader.conf console=comconsole,vidconsole

echo 'donttelltheauditors' | pw usermod root -h 0
echo 'donttelltheauditors' | pw useradd isc -g wheel -h 0
```

Then do one final reboot, and let Puppet bring the box back into the fold!

Upgrades take-aways

- As of now, all but one of our classic nodes has been upgraded to modern hardware with proper remote management.
- But because the base layout of FreeBSD is so rock-solid and predictable (i.e. the distro is just two tarballs), it gave us a way to un-paint ourselves from the corner.
- MFSBSD rocks, but it’s only on mm’s little side page, not really in the handbook, not sure why?
- This has me thinking about the future of memory-based operating systems.

Sysadmin Story Number 2: Service Management

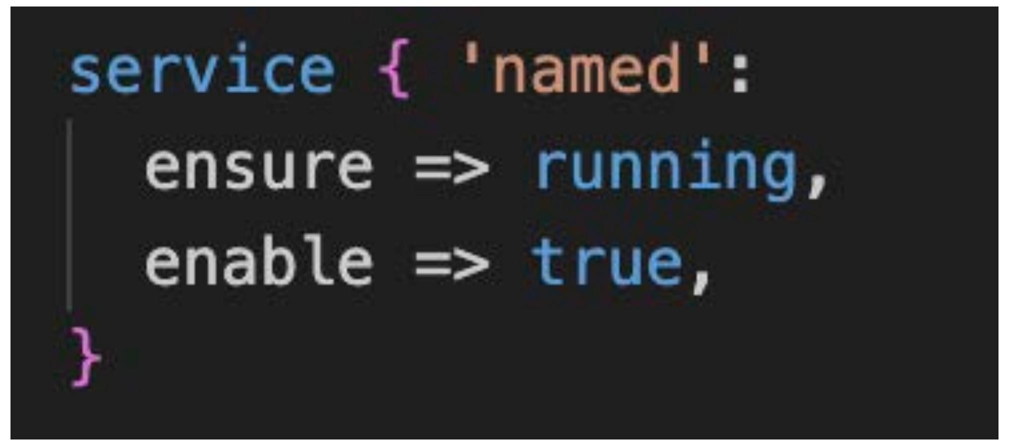
- Most of what I say here for Puppet could be true for Ansible or Chef as well.
- Puppet deploys a service, and services should be modular.
- While puppet does have the ability to “concatenate” a config file together, or build it from templates…
- Services managed by puppet should drop their config files into “dot-d” directories, instead of managing a monolithic file like rc.conf, which extends to files like /etc/rc.conf.d/ syslog.d/, /etc/logrotate.d/, and /usr/local/etc/rc.d/
- An example service…

Service Management: announced

- We also have a shell script running in a loop with daemon(1) which does a dig @localhost . NS
- If the process returns cleanly, the service adds 192.5.5.241 (and the v6 equivalent 2001:500:2f::f) to lo1
- Otherwise, it removes these blocks from lo1
- BIND listens on these new IP addresses
- BIRD notices that these blocks are in the routing table, and adds an aggregate route to its BGP announcement out to all peers.
- We have “break-glass” files in /var that allow us to suppress these announcements if a system is misbehaving for some reason, and the global route will take over.
- As mentioned earlier, proper startup script with shutdown knobs so the script cleans up after itself on a shutdown.

Service Management

- Our services have dependencies, so any service we wrote has a proper rc.d script using the above knobs, so it respects puppet's service "enabled" knob, and "running" knob, which knows standard service commands.



```
service { 'named':
  ensure => running,
  enable => true,
}
```

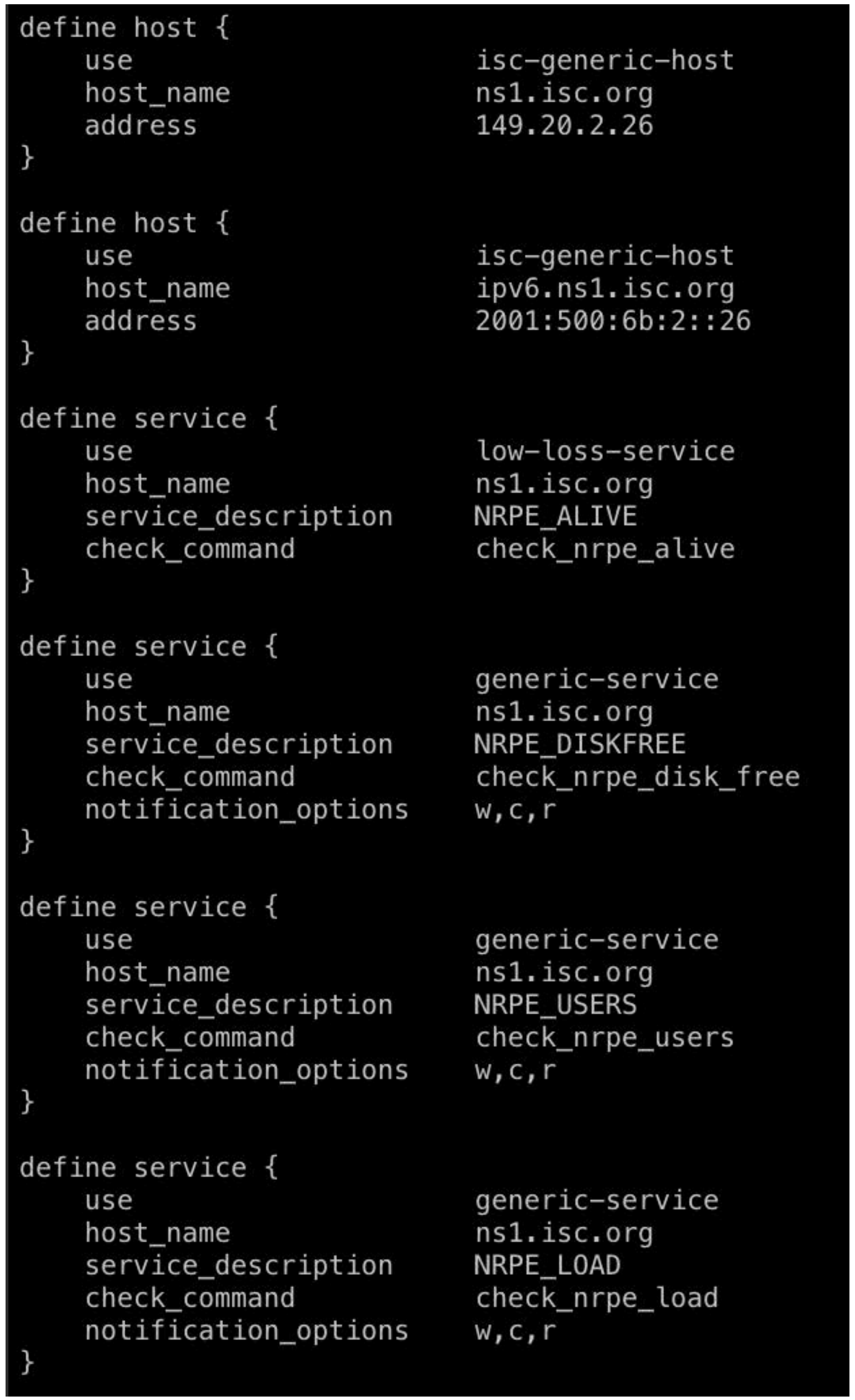
- Which meant we also needed to learn how to properly write things with rcorder(8)
- The handbook's documentation on how services work is a 9/10, and rc.subr is very readable (mostly), the other 1/10 was "just try and and find out".
- Any given service should be able to report "status" with a proper PID file.
- All our services pass sysutils/rclint as well, which most base rc.d files don't.
- Puppet also ties into Monitoring. I have stories about Monitoring.

So I’ve gotten good at writing Nagios plugins

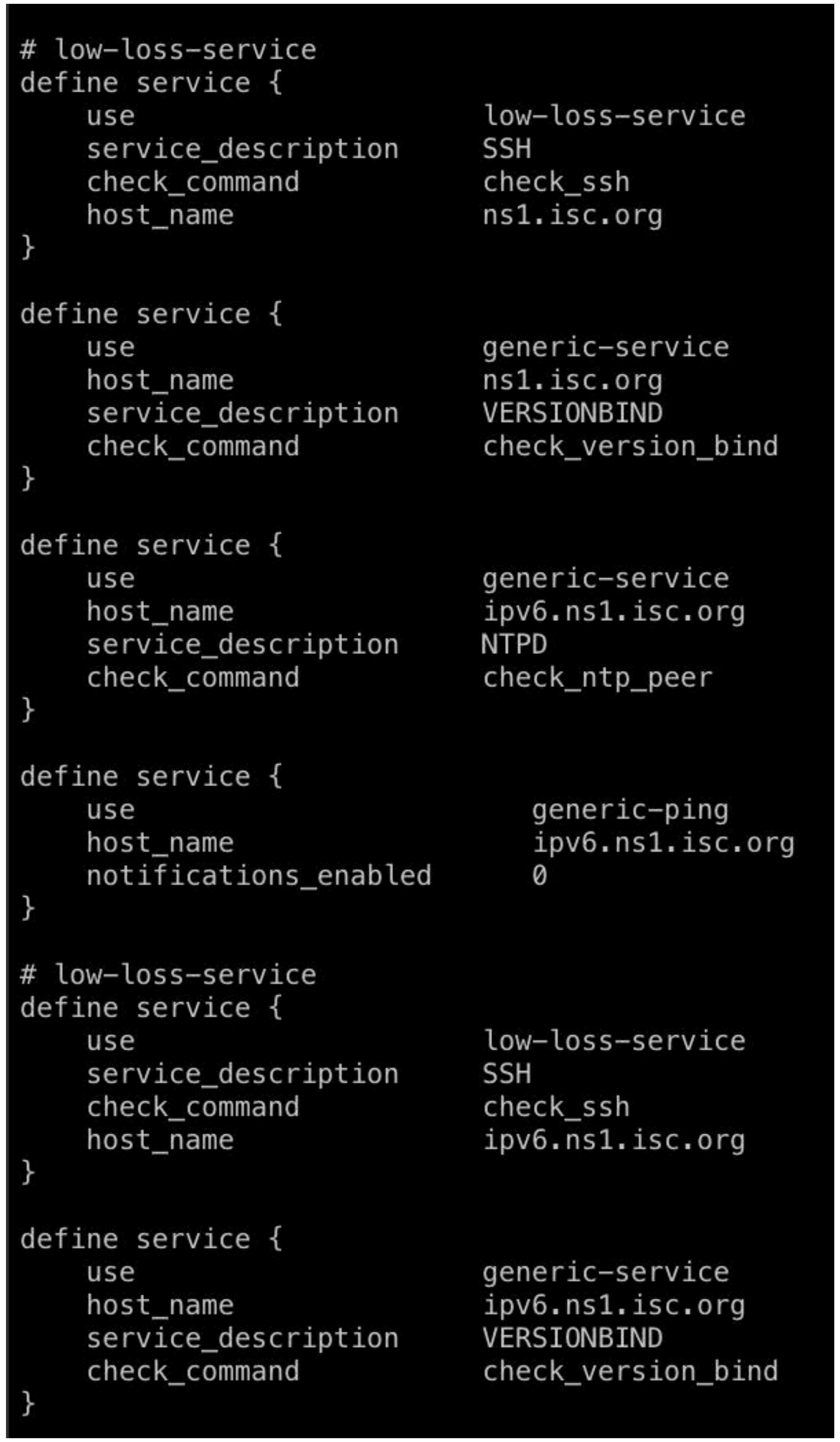
- Plugins to interrogate Liebert UPSes that only speak snmpv1 (not even telnet)
- Tools that go through in order and interrogate tw_cli, sa_cli, mfiutil, mptutil, mrsasutil, megacli, zfs, gmirror, to eventually glean the state of a RAID card, however it was installed “at the time”.
- Tools to talk to old APC UPSes
- Tools to look at the state of our transfer switch. (They wouldn’t let me actually start attaching probes to the engine).
- A number of Environmental monitoring devices.
- And tools to talk to a Dell iDRAC, and look at one OID to see if it’s “okay”, and if not, drill down.
- …did you know that some devices will crash if you monitor them too much? The answer is caching. (You don’t want to hammer a dying raid card with status requests).

Monitoring with Puppet

- A puppet-deployed service can also deploy things like /usr/local/etc/nrpe.d/ files, so nagios/nrpe can check services on the local machine, for services that don't expose a network port, in addition to the core nrpe.cfg
- Puppet also adds virtual resources, which our nagios system picks up — any box running a service will have a corresponding health-check added on our nagios systems Adding include isc_monitor::servicename to a service causes nothing extra to be deployed on the host.
- Puppet can expose "facts" — intrinsic details about a system, and we do things like expose ilom IP's as custom facts.
- But on the next puppet run on a monitoring box, files will be deployed in /usr/local/etc/nagios/objects/hostname
- Which are a collection of generated templates, concatenated into a single per-host-file
- If it's a *physical* box, we grab the iLOM/iDRAC IP via ipmitool and build monitoring for that as well.

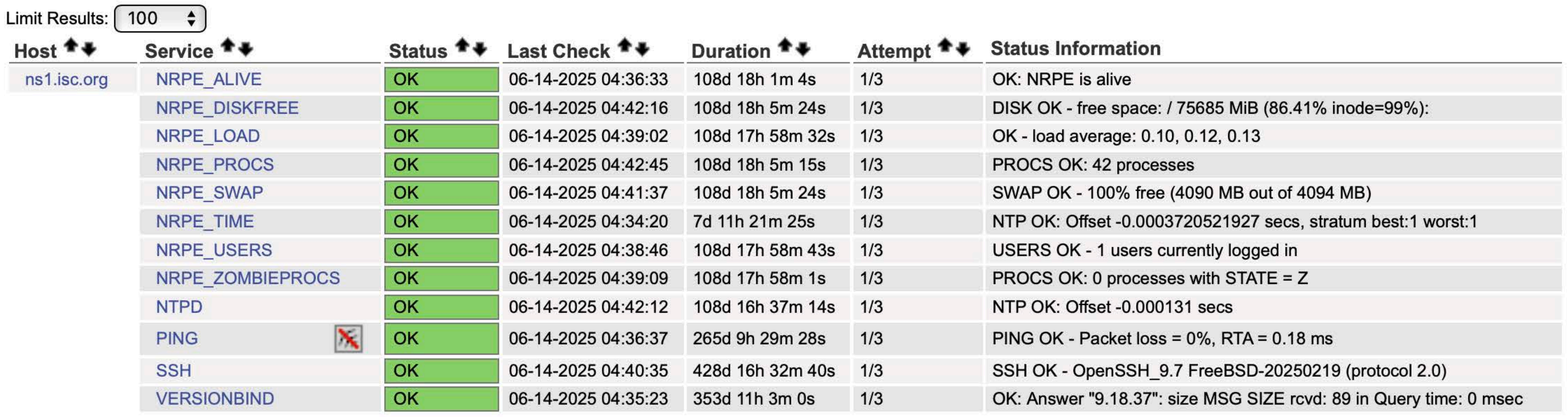


```
define host {
    use                         isc-generic-host
    host_name                   ns1.isc.org
    address                     149.20.2.26
}

define host {
    use                         isc-generic-host
    host_name                   ipv6.ns1.isc.org
    address                     2001:500:6b:2::26
}

define service {
    use                         low-loss-service
    host_name                   ns1.isc.org
    service_description         NRPE_ALIVE
    check_command               check_nrpe_alive
}

define service {
    use                         generic-service
    host_name                   ns1.isc.org
    service_description         NRPE_DISKFREE
    check_command               check_nrpe_disk_free
    notification_options        w,c,r
}

define service {
    use                         generic-service
    host_name                   ns1.isc.org
    service_description         NRPE_USERS
    check_command               check_nrpe_users
    notification_options        w,c,r
}

define service {
    use                         generic-service
    host_name                   ns1.isc.org
    service_description         NRPE_LOAD
    check_command               check_nrpe_load
    notification_options        w,c,r
}
```



```
# low-loss-service
define service {
    use                         low-loss-service
    service_description         SSH
    check_command               check_ssh
    host_name                   ns1.isc.org
}

define service {
    use                         generic-service
    host_name                   ns1.isc.org
    service_description         VERSIONBIND
    check_command               check_version_bind
}

define service {
    use                         generic-service
    host_name                   ipv6.ns1.isc.org
    service_description         NTPD
    check_command               check_ntp_peer
}

define service {
    use                            generic-ping
    host_name                      ipv6.ns1.isc.org
    notifications_enabled          0
}

# low-loss-service
define service {
    use                         low-loss-service
    service_description         SSH
    check_command               check_ssh
    host_name                   ipv6.ns1.isc.org
}

define service {
    use                         generic-service
    host_name                   ipv6.ns1.isc.org
    service_description         VERSIONBIND
    check_command               check_version_bind
}
```

Nagios Screenshot



Limit Results: 100

Host	Service	Status	Last Check	Duration	Attempt	Status Information
ns1.isc.org	NRPE_ALIVE	OK	06-14-2025 04:36:33	108d 18h 1m 4s	1/3	OK: NRPE is alive
	NRPE_DISKFREE	OK	06-14-2025 04:42:16	108d 18h 5m 24s	1/3	DISK OK - free space: / 75685 MiB (86.41% inode=99%):
	NRPE_LOAD	OK	06-14-2025 04:39:02	108d 17h 58m 32s	1/3	OK - load average: 0.10, 0.12, 0.13
	NRPE_PROCS	OK	06-14-2025 04:42:45	108d 18h 5m 15s	1/3	PROCS OK: 42 processes
	NRPE_SWAP	OK	06-14-2025 04:41:37	108d 18h 5m 24s	1/3	SWAP OK - 100% free (4090 MB out of 4094 MB)
	NRPE_TIME	OK	06-14-2025 04:34:20	7d 11h 21m 25s	1/3	NTP OK: Offset -0.0003720521927 secs, stratum best:1 worst:1
	NRPE_USERS	OK	06-14-2025 04:38:46	108d 17h 58m 43s	1/3	USERS OK - 1 users currently logged in
	NRPE_ZOMBIEPROCS	OK	06-14-2025 04:39:09	108d 17h 58m 1s	1/3	PROCS OK: 0 processes with STATE = Z
	NTPD	OK	06-14-2025 04:42:12	108d 16h 37m 14s	1/3	NTP OK: Offset -0.000131 secs
	PING	OK	06-14-2025 04:36:37	265d 9h 29m 28s	1/3	PING OK - Packet loss = 0%, RTA = 0.18 ms
	SSH	OK	06-14-2025 04:40:35	428d 16h 32m 40s	1/3	SSH OK - OpenSSH_9.7 FreeBSD-20250219 (protocol 2.0)
	VERSIONBIND	OK	06-14-2025 04:35:23	353d 11h 3m 0s	1/3	OK: Answer "9.18.37": size MSG SIZE rcvd: 89 in Query time: 0 msec

Net Results

- As soon as you deploy any host, you do pkg install puppet8; puppet agent -tv
    - You get a base level config with proper sshd settings
    - proper users set up and added to .k5login
    - puppet adds the system to our kerberos domain
    - base level monitoring, plus monitoring for any service deployed via puppet (without ever having to edit a file).
    - And firewalling. Since many of our systems are a solo box out in the world on a public IP, this is key.

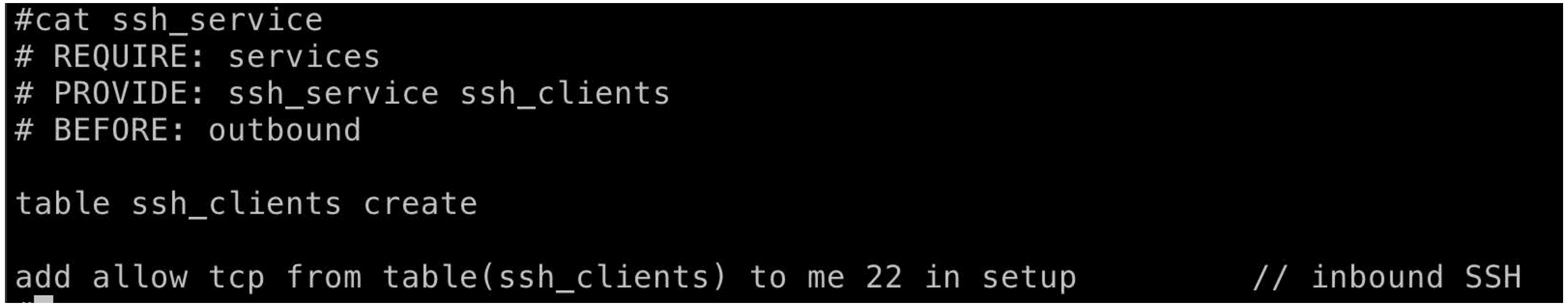
Combat Story 2b: Firewall Startup

- You can set firewall_type in rc.conf, which, if set to a filename, will let you list the name of a file with rules to run (without the “ipfw” word at the beginning).
- What if that file was a directory of files, that were able to be parsed with rcorder, so that any service deployed by puppet, could also deploy ipfw rules for that service?
- I’ve had a PR in for a few years, which we’d love to see mainlined, but for the moment, we override firewall_script for a copy that runs rcorder over /etc/ipfw.d/files and runs them in order.
- We really think that with the dot-d’ification of other tools, this is a natural fit for base (we’d be happy to give up our files that replicate the base examples).

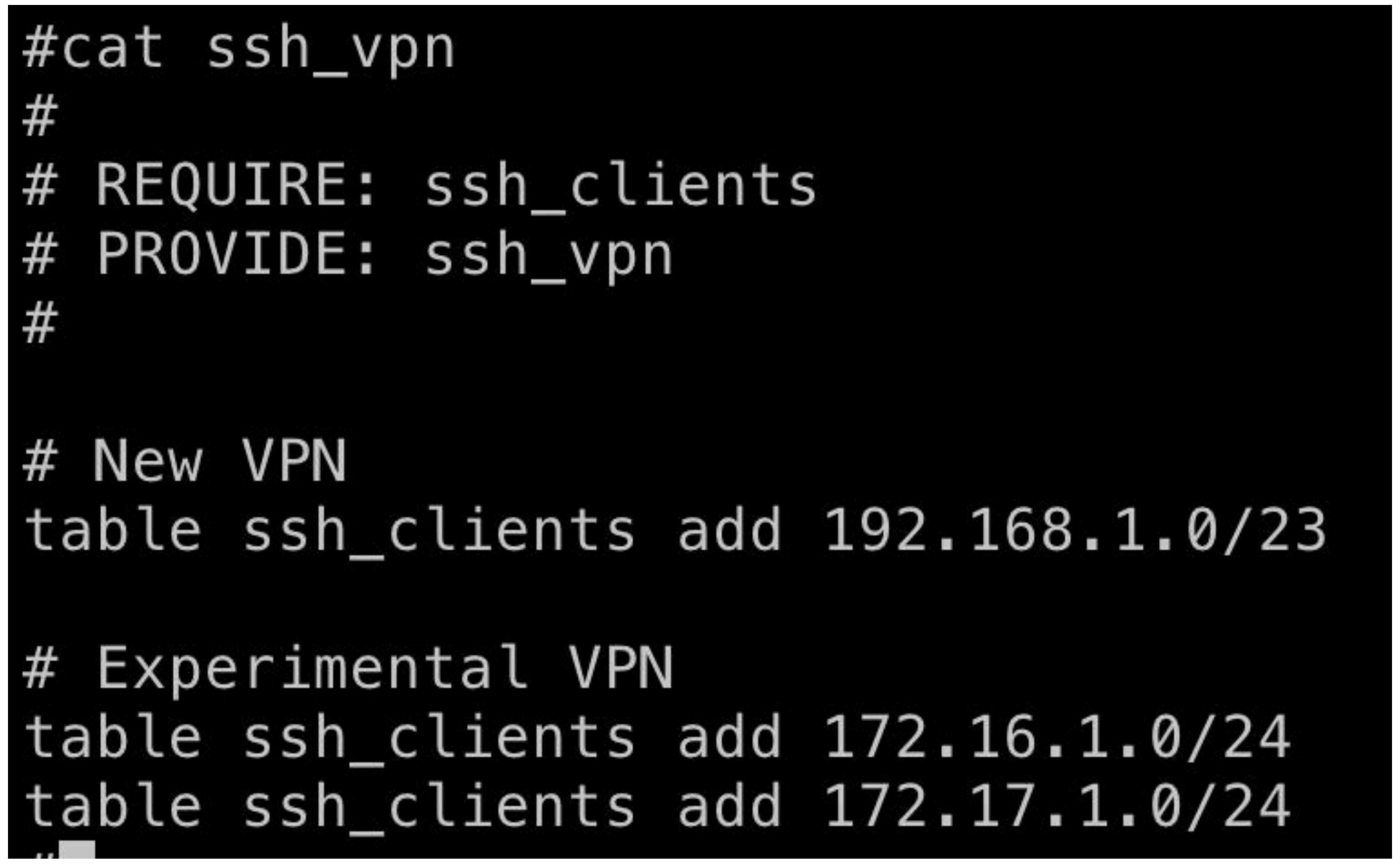
Some ipfw.d examples

- Meta-classes: outbound, routing, services
- Setup and Final rules that add things like core allow lo0 rules and default icmp rules (just like the example rules in rc.firewall).
- Any firewall rule which depends on a table either creates a named table, or has an rcorder rule which requires the table be created first
  - For example, a file that creates the service (ssh_service, but also PROVIDES the ssh_clients table)
  - And then files (ssh_vpn, ssh_monitorhosts, ssh_corporate_shell_server, or even ssh_from_all) that require and which affect that table.
- There’s a “local” file that is added by puppet, but which gives us a place to put local overrides while testing.
- Since all network-facing services in puppet install an ipfw rule, they all have “ipfw” as a dependency, an install of a new service will cause a notification to do a service reload of ipfw.

Some ipfw.d screenshots (edited)



```
#cat ssh_service
# REQUIRE: services
# PROVIDE: ssh_service ssh_clients
# BEFORE: outbound

table ssh_clients create

add allow tcp from table(ssh_clients) to me 22 in setup         // inbound SSH
```



```
#cat ssh_vpn
#
# REQUIRE: ssh_clients
# PROVIDE: ssh_vpn
#

# New VPN
table ssh_clients add 192.168.1.0/23

# Experimental VPN
table ssh_clients add 172.16.1.0/24
table ssh_clients add 172.17.1.0/24
```

Combat Story #3: Dell DRM

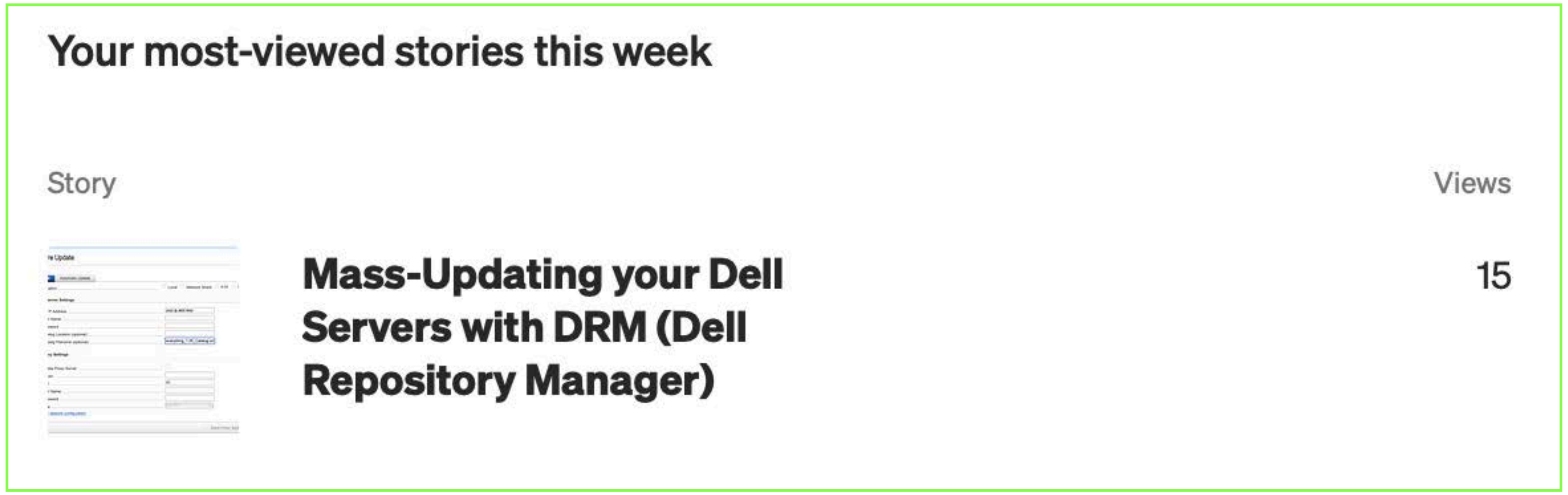
- We ask our patrons to get Dell hardware, but because of in-country differences, we often get a mess of what's available via which reseller at which time. Sometimes we get Weird Things.
- For example, a Dell-branded 10G network card that would put out LLDP packets that were invisible from tcpdump. (In our case, it was connected to an Internet Exchange, which are normally pretty strict about broadcast packets).
- The fix was a firmware update, and we had a deployment method.
- DRM in this case is "Dell Repository Manager".
- This is a tool to fetch all updates from a Dell server and put them someplace your dell iDRAC's can reach them, ala Windows Update.
- If updates are there, the iDRAC will (when commanded) stage them, cleanly shut your system down, boot into an installer, and then boot back into your OS.

More on DRM

- It’s a Java (command-line and GUI) app that requires *sigh* Linux, so we have a necessary Linux control plane for our Dell stuff.

- You can build a repository for a model of machine or bunch of models, then deploy those to a web accessible directory.

- You can make the idrac check that repo for available updates with either a click in the iDRAC gui, or can run a check over ssh, which you can later poll the results of.



Your most-viewed stories this week

Story	Views
Mass-Updating your Dell Servers with DRM (Dell Repository Manager)	15

While we’re talking about Dell

- There’s also a Dell tool called racadm, which talks to the iDRAC and do things you can’t do via ssh or redfish
- For example, you need racadm to update the SSL certificate programmatically (we have our own CA, but CA/Browser forum rules still mean we need to refresh it every two years).
- Racadm *does* run under the FreeBSD Linuxulator, and there’s a PR for a port (originally contributed by Norse), but the license makes it a bit of a nightmare, I could use guidance here.
- I’ve also attempted to do some probing into what it would take to run an iDrac “service module” under FreeBSD, which would let the iDrac log events to the underlying OS.
- The idrac exposes an internal USB ethernet visible to the host OS for IPC — I’d love to document this some more, bringing us closer to an open-source iSM equivalent.

Future Work (security and reliability)

- At the end of the day, we’re a small team running a critical internet service on commodity hardware, often in places with limited connectivity back to our control plane (but that’s the point!)
- We’ve leveraged an OS that’s easy to work with to tie our deployment tools into it, but that hardware is still far away and not in our hands.
- Every system we deploy has unique passwords, so compromising one would not gain an attacker much. We do monitor for unexpected reboots and chassis intrusions, but some of our sites have really spotty power.
- Would my past fun with MFSBSD make those systems more resilient if they could boot up read-only?
- I’ve considered trying to find a way to do encrypted storage that would depend on a TPM attestation in order to boot cleanly, but haven’t found anything useful. (There was a TPM talk at BSDCan a few years ago, with work done by Juniper).
- That said, these are not Netflix-class machines full of copyrighted content — the data these are serving is public data for the good of the internet.
- The contents of the queries is precious, but there are easier ways to get that than compromising our OS. (DNS is still a plaintext protocol, at least at the root level).

Future Work (DNS Resiliency)

- The DNS continues to be a service of growing complexity.
- For resiliency reasons, ISC prefers to run root services only on bare-metal, to prevent supply chain attacks that might be present in container-stacks. (We've made a few minor exceptions for deployments in active combat zones).
- We've also experimented with small deployable boxes, but we don't feel comfortable authorizing anyone to use the F-Root IP address space on a local mirror they control (so we're not about to say "Here, run this RPi image on your own hardware and you can have your own F").
- This is critical infrastructure, it shouldn't be exciting. Stability in the face of disasters is the story I'm hoping to tell.

Questions?

Notes (added post-talk)

- My medium blog on Dell DRM: https://gushi.medium.com/mass-updating-your-dell-servers-with-drm-dell-repository-manager-8b46e78614c5
(apologies for the Medium paywall if you hit it -- I'm working on converting to a non-monetized/paid account -- note that my medium blog contains non-work items and any opinions there are my own and do not represent my employer)
- Historic ISC RFC-like document on Anycast (circa 2003): https://www.isc.org/pubs/tn/isc-tn-2003-1.html
- If you're interested in hosting an F-Root server (if you're an IXP or ISP): https://www.isc.org/froot-process/
- Martin Matuška's MFSBSD Page: https://mfsbsd.vx.sk/
(I was told after the talk that these are likely to be supported as official repos)
- My bug report to support ipfw rules in a directory: https://bugs.freebsd.org/bugzilla/show_bug.cgi?id=266137