Next-Gen Utilities in RHEL

AKA: stuff I meant to learn a while back

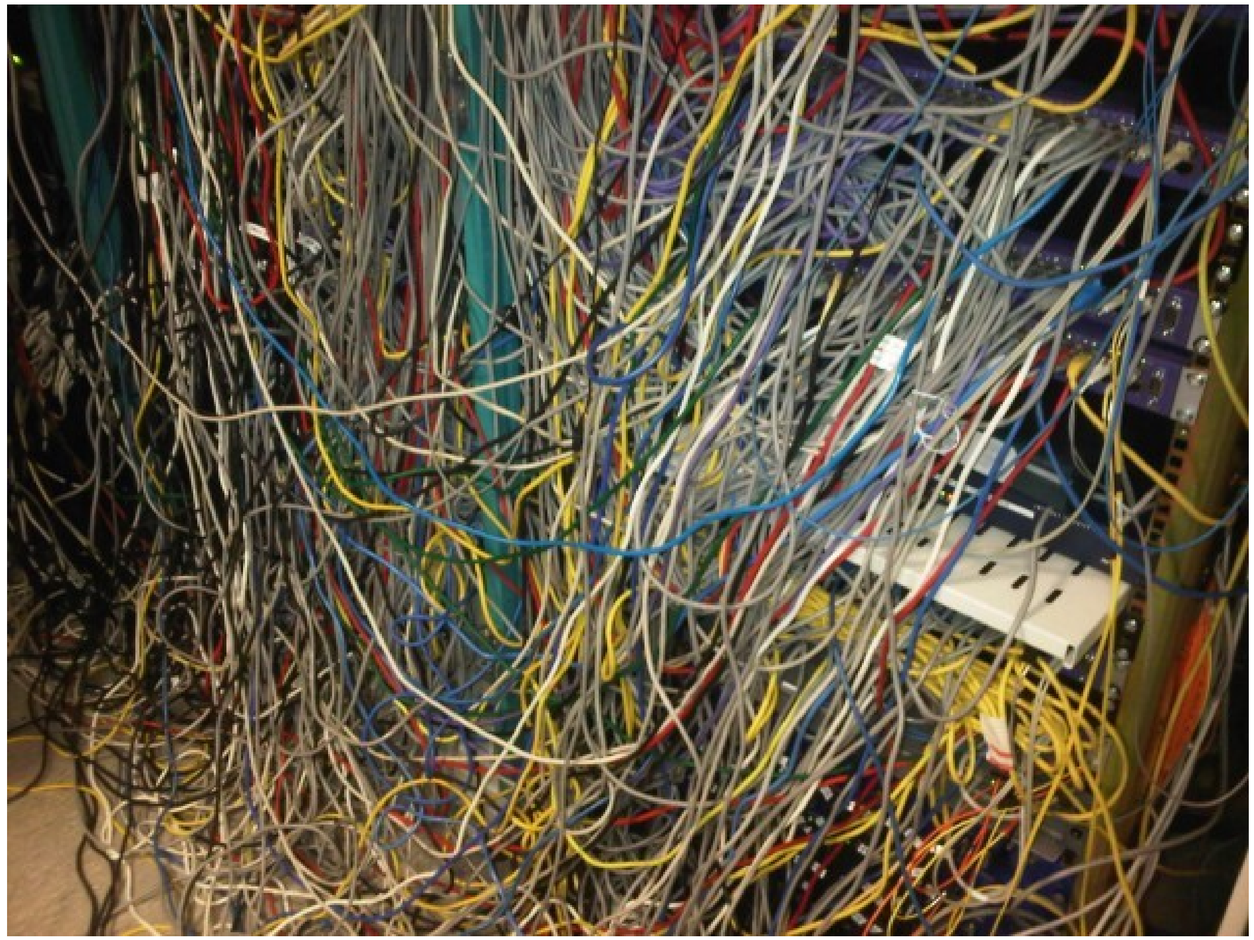
Ben Breard
Solutions Architect, Red Hat

redhat.

Agenda

- Networking
- Storage
- Software Collections
- Discussion: What newer utilities have been valuable in your environment?

Networking



Networking

- NetworkManager – multiple ways to configure
  - Manually edit ifscripts /etc/sysconfig/network-scripts/ifcfg-XXX
  - nmtui
  - nmcli
- Network – The “network service” init script
- networkd – Maybe in the future
  - http://www.freedesktop.org/software/systemd/man/systemd-networkd.service.html

iproute2

iproute2

- Uses netlink socket interface
  - Much lighter weight
- Dates back to 2.2
- Work with namespaces

net-tools

- Uses /procfs and ioctl
  - /proc considered heavy
  - ioctl - obsolete kernel interface
- Doesn't support InfiniBand addresses
- Maintained, but no real future

net-tools

- Still shipped, but not included in @core
- Considered deprecated

This guy uses net-tools

```
# rpm -ql net-tools
/bin/netstat
/sbin/arp
/sbin/ether-wake
/sbin/ifconfig
/sbin/ipmaddr
/sbin/iptunnel
/sbin/mii-diag
/sbin/mii-tool
/sbin/nameif
/sbin/plipconfig
/sbin/route
/sbin/slattach
```

Quick Command Review

Deprecated	Replacement
arp	ip n (ip neighbor)
ifconfig	ip a (ip addr), ip link, ip -s (ip -stats)
iptunnel	ip tunnel
iwconfig	iw
nameif	ip link, ifrename
netstat	ss, ip route (for netstat-r), ip -s link (for netstat -i), ip maddr (for netstat-g)
route	ip r

ip command

- ip command cheat sheet:
  - https://access.redhat.com/sites/default/files/attachments/rh_ip_command_cheatsheet_1214_jcs_print.pdf
    - Print and hang next to your bathroom mirror
- Really simple to use with bash-completion
- NetworkManager and network service aware
- Not persistent

```
# ip link set up ens9 up
# ip addr add 10.10.10.50/24 dev ens9
# ip addr route add default via 10.10.10.254
```

- Bonus: calc and subnetcalc (epel) are your friends

ss – Socket Statistics

- Replacement for netstat
- Getting detailed information is as easy as `-pie`

```
# ss -tmpie
```

```
ss -tmpie
State       Recv-Q Send-Q                                  Local Address:Port                                  Peer Address:Port
ESTAB       0      0                                    192.168.81.25:ssh                                  192.168.81.7:50250    timer:(keepalive,118min,0) users:(("sshd",pid=7287,fd=3))
ino:3943759 sk:ffff88021c29ab80 <->
         skmem:(r0,rb374400,t0,tb165376,f0,w0,o0,bl0) ts sack cubic wscale:7,7 rto:201 rtt:0.189/0.073 ato:40 mss:1448 cwnd:10 ssthresh:18 send 612.9Mbps pacing_rate 1223.4Mbps rcv_rtt:1 rcv_space:28960
```

- Listening on UDP & TCP sockets

```
# ss -lut |column -t
```

```
Netid  State   Recv-Q  Send-Q  Local              Address:Port  Peer  Address:Port
tcp    LISTEN  0       128     *:sunrpc                  *:*
tcp    LISTEN  0       5       10.10.20.254:domain       *:*
tcp    LISTEN  0       5       192.168.122.1:domain      *:*
tcp    LISTEN  0       128     *:ssh                     *:*
tcp    LISTEN  0       128     127.0.0.1:ipp             *:*
tcp    LISTEN  0       50      *:60536                   *:*
```

Ummm yeah soooh **`netstat -tulp`** is now **`ss -tulp`**

Nic Teaming!

Bonding

Pros

- Very stable and mature
- Pervasive usage
- Great performance

Cons

- Bonding driver has a dated architecture
  - Control, management, & data paths are in kernel space
  - Not flexible
- Becoming bloated and difficult to maintain
- Future development? New features?

Uncle Joey prefers bonding

Teaming

- Solves the exact same problems as bonding
- Improved
  - Configuration
  - Management
  - Monitoring
- No loss of features or throughput
- Great performance due to less overhead

Uncle Jesse uses teaming

Teaming & Bonding Performance

Interface	Performance: 64byte packets	Performance: 1KB packets	Performance: 64KB packets	Average Latency
eth0	1664.00Mb/s (27.48%CPU)	8053.53Mb/s (30.71%CPU)	9414.99Mb/s (17.08%CPU)	54.7usec
eth1	1577.44Mb/s (26.91%CPU)	7728.04Mb/s (32.23%CPU)	9329.05Mb/s (19.38%CPU)	49.3usec
Bonded (eth+eth1)	1510.13Mb/s (27.65%CPU)	7277.48Mb/s (30.07%CPU)	9414.97Mb/s (15.62%CPU)	55.5usec
Teamed (eth0+eth1)	1550.15Mb/s (26.81%CPU)	7435.76Mb/s (29.56%CPU)	9413.8Mb/s (17.63%CPU)	55.5usec

Migrating

- bond2team – will migrate ifcfg files from bonds to teams.
  - Note: will not adjust firewall rules, or anything else tied to the old interface name.

```
# /usr/bin/bond2team --master bond0
# /usr/bin/bond2team --master bond0 --rename team0
# /usr/bin/bond2team --bonding_opts "mode=1 miimon=500"
```

teamd

- Start daemon

```
# teamd -t team0 -d
```

- Add ports
  - Note: Make sure the link is down for slave interfaces `ip link set [ensX] down`

```
# teamdctl team0 port add ens9
# teamdctl team0 port add ens10
```

- Enable link & add ip

```
# ip link set team0 up
# ip addr add 10.10.10.50/24 dev team0
```

teamd

```
# teamdctl team0 state
setup:
  runner: roundrobin
ports:
  ens9
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
  ens10
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
```

teamd

- Dump config

```
# teamdctl team0 config dump actual > team0.json
```

- Remove team0

```
# teamd -t team0 -k
```

- **Start team0 from config**

```
# teamd -f team0.json -d
```

- **Or you can cheat!**

```
/usr/share/doc/teamd-1.15/example_configs/
/usr/share/doc/teamd-1.15/example_ifcfgs/
```

Teaming ifcfg

```
DEVICE=team0
DEVICETYPE=Team
ONBOOT=yes
BOOTPROTO=none
IPADDR=192.168.11.1
PREFIX=24
TEAM_CONFIG='{"runner": {"name": "activebackup"}, "link_watch":
{"name": "ethtool"}}'
```

```
DEVICE=ens1
HWADDR=XX:XX:XX:XX:XX:XX
DEVICETYPE=TeamPort
ONBOOT=yes
TEAM_MASTER=team0
TEAM_PORT_CONFIG='{"prio": 100}'
```

Storage

fdisk < parted

fdisk

- Great utility
- Everyone knows it
- Fails w/ large luns (>= 2TB)
- Meh GPT support

parted

- Great utility
- A lot of people know it
- Succeeds with large luns!!
- GPT support!!!
- Natively scriptable

Parted Usage

```
# parted -s /dev/sdc mkabel gpt
# parted -s /dev/sdc mkpart primary 0% 2g
# parted -s /dev/sdc mkpart primary 2g 4g
# parted -s /dev/sdc mkpart primary 6g 100%
# parted /dev/sdc p
```

```
Model: ATA TS256GMTS400 (scsi)
Disk /dev/sdb: 256GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name     Flags
 1      1049kB  2000MB  1999MB               primary
 2      2000MB  4000MB  2001MB               primary
 3      6000MB  8589MB  2589MB               primary
```

System Storage Manager (ssm)

System Storage Manager (ssm)

- Manage advanced features on multiple backends (lvm, btrfs, crypt)
- Included in Anaconda --> available on tty2
- `yum install system-storage-manager`
- **Devices** - devices become building blocks for more advanced storage setups. e.g. SATA, SCSI, virtio, etc
- **Pools** - Set of grouped devices in a logical volume.
  - lvm backend or btrfs root volume.
- **Volumes** - Final volume constructed.
- **Snapshots** - Represents lvm and/or btrfs snapshot volumes/subvolumes.

...before

...before

- `pvcreate /dev/sd{b,c}`

...before

- `pvcreate /dev/sd{b,c}`
- `vgcreate [vgname] /dev/sd{b,c}`

...before

- `pvcreate /dev/sd{b,c}`
- `vgcreate [vgname] /dev/sd{b,c}`
- `lvcreate -L 120G -n [lvname]`

...before

- `pvcreate /dev/sd{b,c}`
- `vgcreate [vgname] /dev/sd{b,c}`
- `lvcreate -L 120G -n [lvname]`
- `mkfs.ext4 /dev/[vgname]/lvname]`

...before

- `pvcreate /dev/sd{b,c}`
- `vgcreate [vgname] /dev/sd{b,c}`
- `lvcreate -L 120G -n [lvname]`
- `mkfs.ext4 /dev/[vgname]/lvname]`
- `mount /dev/mapper/vgname-lvname /mnt/data`

...before

- `pvcreate /dev/sd{b,c}`
- `vgcreate [vgname] /dev/sd{b,c}`
- `lvcreate -L 120G -n [lvname]`
- `mkfs.ext4 /dev/[vgname]/lvname]`
- `mount /dev/mapper/vgname-lvname /mnt/data`

.....wait for it

...before

- `pvcreate /dev/sd{b,c}`
- `vgcreate [vgname] /dev/sd{b,c}`
- `lvcreate -L 120G -n [lvname]`
- `mkfs.ext4 /dev/[vgname]/lvname]`
- `mount /dev/mapper/vgname-lvname /mnt/data`

.....wait for it

...after

- `ssm create -s 120G --fstype ext4 /dev/sd{b,c} /mnt/data`

Software Collections

....the most underrated thing that we ship

Software Collections

Have you ever thought:

- “I wish RHEL X.x shipped with a newer version of Y?”
- “Bahhh! These developers always want the latest and greatest, but they don't deal with the maintenance and patching.”
- “Maybe I'll just grab the latest bits, but what about support, rpm dependencies, etc”

Software Collections

- Allow newer versions of languages, frameworks, and databases to be installed in parallel with packages shipped in RHEL.
- Developers & admins get what they want
- We keep our API/ABI guarantee
- Include tools to build your own
- Everybody wins

Software Collections 1.2 – RHEL 6 & 7 (rpms + docker files)

- Perl 5.16.3
- PHP 5.4.16
- PHP 5.5.6
- Python 2.7.5
- Python 3.3.2
- Ruby 1.9.3
- Ruby 2.0.0
- Ruby on Rails 4.0.2
- MariaDB 5.5.37
- MongoDB 2.4.9
- MySQL 5.5.37
- PostgreSQL 9.2.8
- Node.js 0.10
- Nginx 1.6.1
- Apache httpd 2.4.6
- Thermostat 1.0.4
- Git 1.9.4
- DevAssistant 0.9.1
- Maven 3.0.5

Software Collections - Usage

- Included with RHEL
- Enable: rhel-variant-rhscl-6-rpms (RHN: rhel-x86_64-variant-6-rhscl-1)
  - optional channel required for some packages
- Leveraging SCL:

```
# scl enable software_collection... 'command...'
# scl enable perl516 'perl hello.pl'
```

- Default to SCL in a shell

```
# scl enable python33 bash
```

- Services work the same: `service postgresql92-postgresql start`

What else is working for you guys?


redhat.