ubuCon ASIA 2021

Using kexec to speed-up reboot

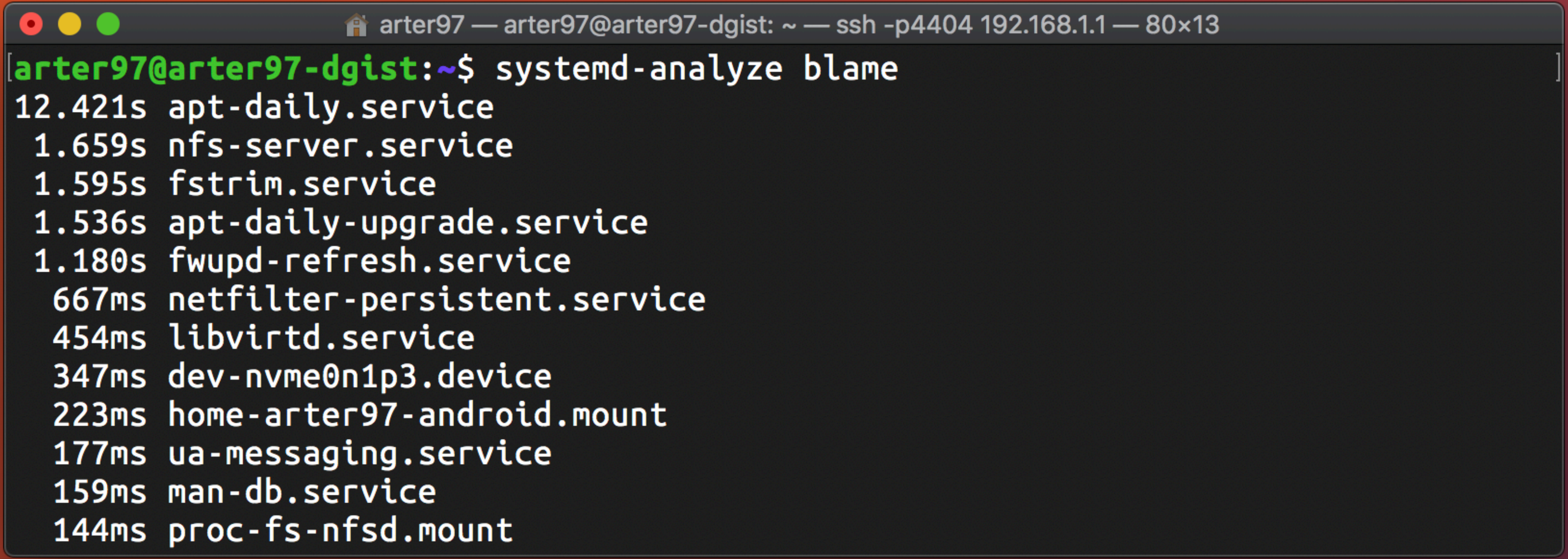
Juhyung Park

arter97@dgist.ac.kr

arter97

@arter97

Standard practices to reduce boot time

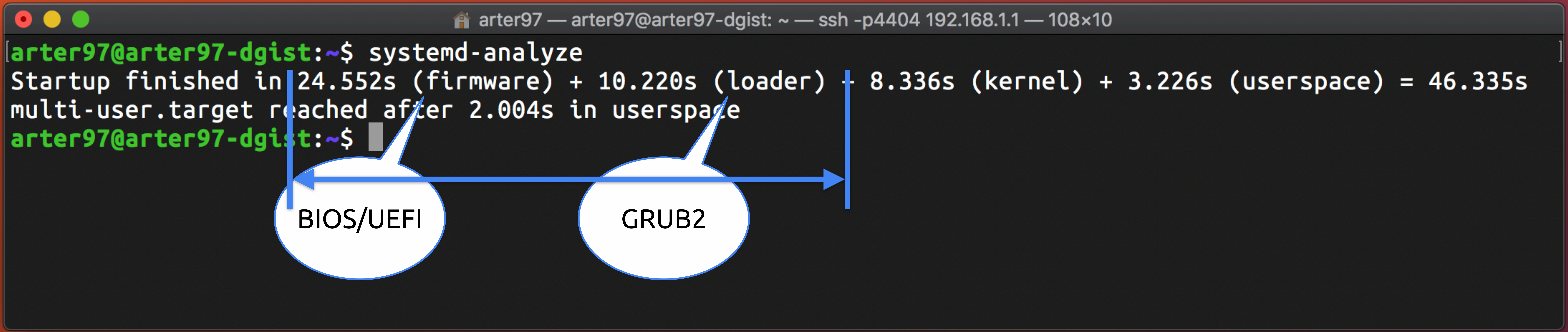


arter97 — arter97@arter97-dgist: ~ — ssh -p4404 192.168.1.1 — 80×13

```
arter97@arter97-dgist:~$ systemd-analyze blame
12.421s apt-daily.service
 1.659s nfs-server.service
 1.595s fstrim.service
 1.536s apt-daily-upgrade.service
 1.180s fwupd-refresh.service
  667ms netfilter-persistent.service
  454ms libvirtd.service
  347ms dev-nvme0n1p3.device
  223ms home-arter97-android.mount
  177ms ua-messaging.service
  159ms man-db.service
  144ms proc-fs-nfsd.mount
```

```
$ systemd-analyze blame
$ systemd-analyze plot > plot.svg
```

Reboots are still slow



```
arter97@arter97-dgist:~$ systemd-analyze
Startup finished in 24.552s (firmware) + 10.220s (loader) + 8.336s (kernel) + 3.226s (userspace) = 46.335s
multi-user.target reached after 2.004s in userspace
arter97@arter97-dgist:~$
```

arter97 — arter97@arter97-dgist: ~ — ssh -p4404 192.168.1.1 — 108×10
BIOS/UEFI
GRUB2

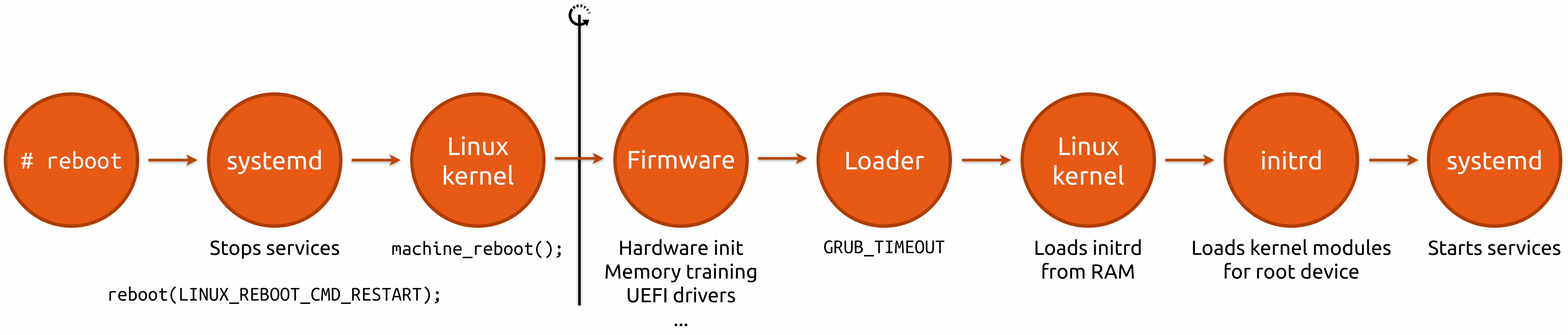
Firmware and loader accounts 75%

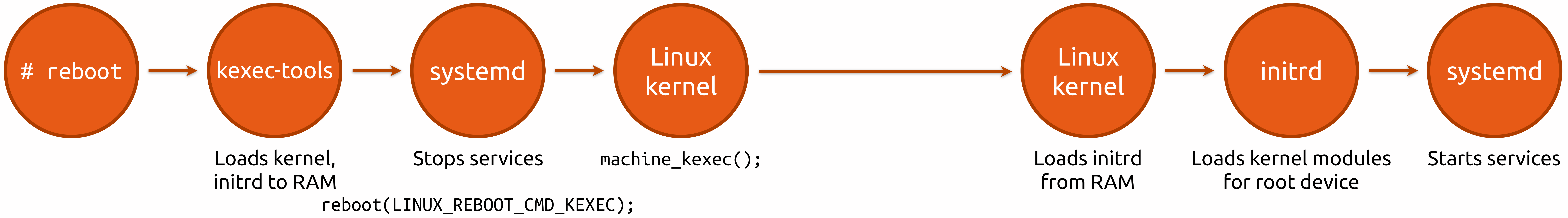
Things get worse for workstations/servers

Enter kexec

kexec can bypass the firmware and loader for reboots

Regular reboots vs. kexec





How to use kexec on Ubuntu (1/2)

- `apt install kexec-tools`
- `apt remove finalrd`
  - With finalrd, systemd fails to find kexec binary and performs a full reboot instead
  - ubuntu-server installs mdadm, which installs finalrd - mdadm still works without finalrd
  - This was fixed (workaround) in systemd v246 - Ubuntu 20.04 ships with systemd v245 :(
  - ... if you really need finalrd, add a new rule to /usr/share/finalrd/kexec.finalrd

How to use kexec on Ubuntu (2/2)

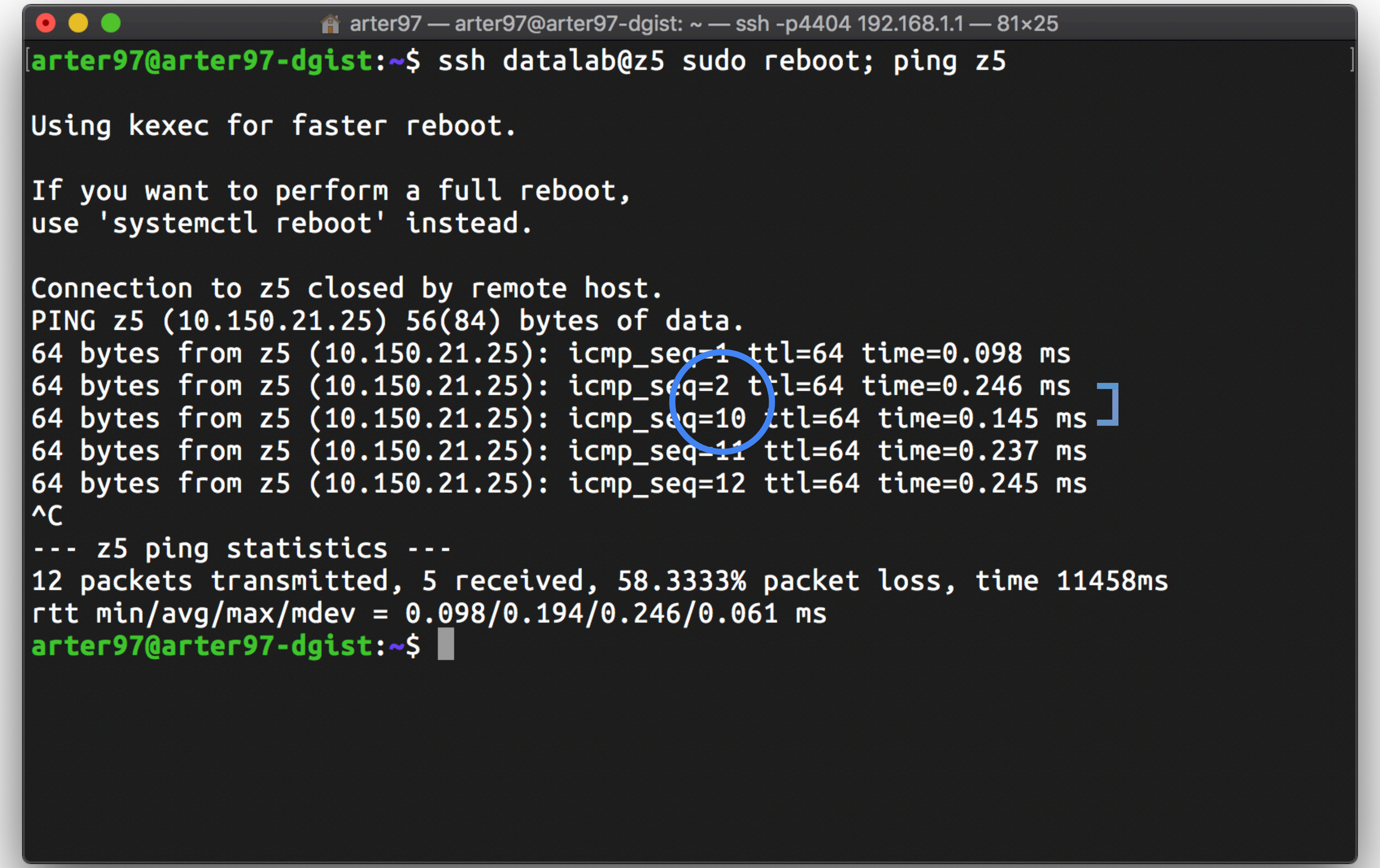
- Custom reboot helper

```bash
#!/bin/bash

# Store at /usr/local/sbin/reboot with 755 permission

echo ""
echo "Using kexec for faster reboot."
echo ""
echo "If you want to perform a full reboot,"
echo "use 'systemctl reboot' instead."
echo ""

sudo kexec \
    -l $(ls /boot/vmlinuz-* | sort -V | tail -n1) \
    --initrd=$(ls /boot/vmlinuz-* | sort -V | tail -n1 | sed s/vmlinuz/initrd.img/g) \
    --reuse-cmdline && sudo systemctl kexec
```

- This script loads the latest kernel and initrd to the RAM and calls systemd for kexec reboot, reusing cmdline

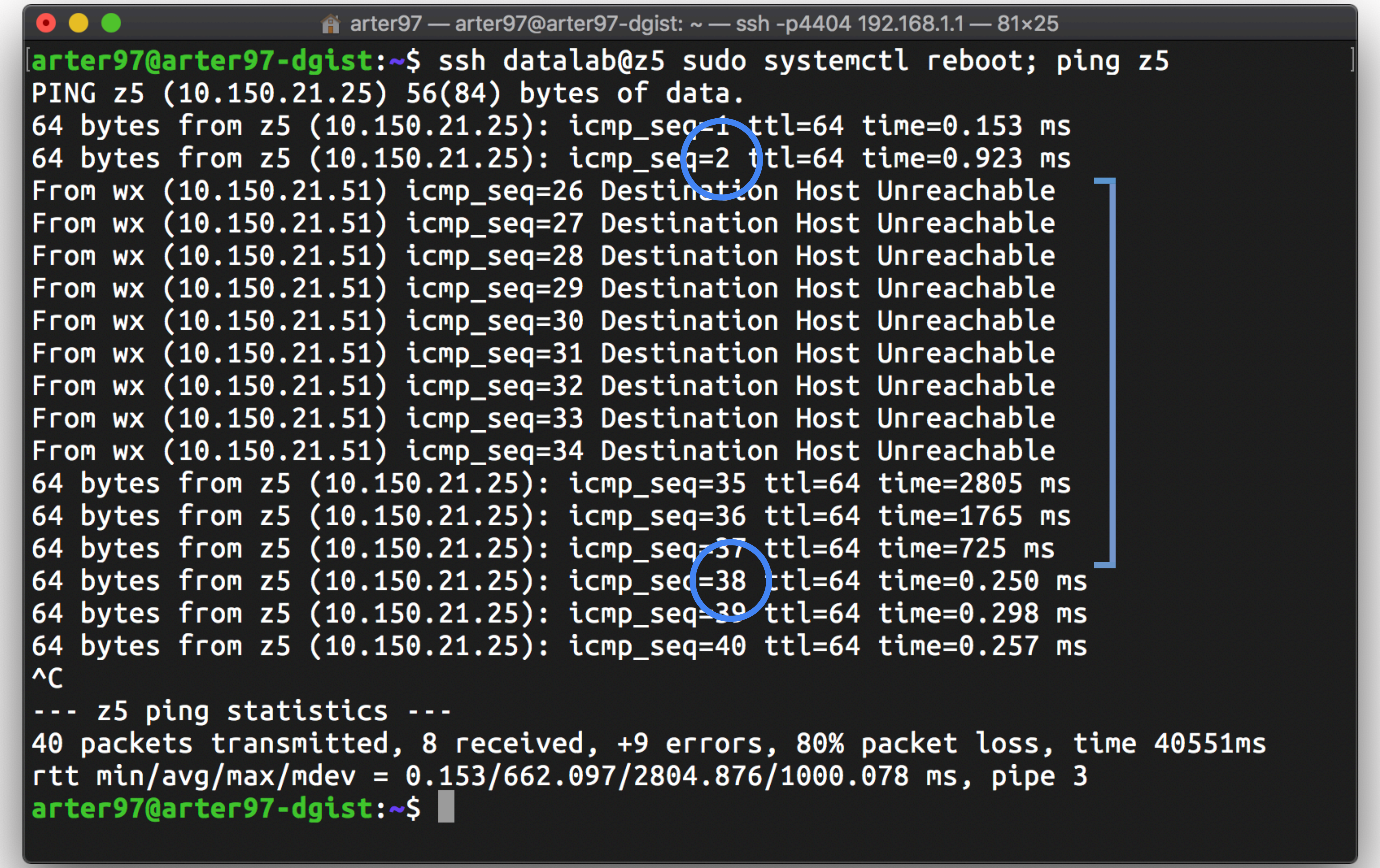
- https://gist.github.com/arter97

How effective is kexec? - Testing how long it takes for the network to get back online


arter97 — arter97@arter97-dgist: ~ — ssh -p4404 192.168.1.1 — 81×25

```
arter97@arter97-dgist:~$ ssh datalab@z5 sudo reboot; ping z5

Using kexec for faster reboot.

If you want to perform a full reboot,
use 'systemctl reboot' instead.

Connection to z5 closed by remote host.
PING z5 (10.150.21.25) 56(84) bytes of data.
64 bytes from z5 (10.150.21.25): icmp_seq=1 ttl=64 time=0.098 ms
64 bytes from z5 (10.150.21.25): icmp_seq=2 ttl=64 time=0.246 ms
64 bytes from z5 (10.150.21.25): icmp_seq=10 ttl=64 time=0.145 ms
64 bytes from z5 (10.150.21.25): icmp_seq=11 ttl=64 time=0.237 ms
64 bytes from z5 (10.150.21.25): icmp_seq=12 ttl=64 time=0.245 ms
^C
--- z5 ping statistics ---
12 packets transmitted, 5 received, 58.3333% packet loss, time 11458ms
rtt min/avg/max/mdev = 0.098/0.194/0.246/0.061 ms
arter97@arter97-dgist:~$
```

▲ kexec reboots in 8s


arter97 — arter97@arter97-dgist: ~ — ssh -p4404 192.168.1.1 — 81×25

```
arter97@arter97-dgist:~$ ssh datalab@z5 sudo systemctl reboot; ping z5
PING z5 (10.150.21.25) 56(84) bytes of data.
64 bytes from z5 (10.150.21.25): icmp_seq=1 ttl=64 time=0.153 ms
64 bytes from z5 (10.150.21.25): icmp_seq=2 ttl=64 time=0.923 ms
From wx (10.150.21.51) icmp_seq=26 Destination Host Unreachable
From wx (10.150.21.51) icmp_seq=27 Destination Host Unreachable
From wx (10.150.21.51) icmp_seq=28 Destination Host Unreachable
From wx (10.150.21.51) icmp_seq=29 Destination Host Unreachable
From wx (10.150.21.51) icmp_seq=30 Destination Host Unreachable
From wx (10.150.21.51) icmp_seq=31 Destination Host Unreachable
From wx (10.150.21.51) icmp_seq=32 Destination Host Unreachable
From wx (10.150.21.51) icmp_seq=33 Destination Host Unreachable
From wx (10.150.21.51) icmp_seq=34 Destination Host Unreachable
64 bytes from z5 (10.150.21.25): icmp_seq=35 ttl=64 time=2805 ms
64 bytes from z5 (10.150.21.25): icmp_seq=36 ttl=64 time=1765 ms
64 bytes from z5 (10.150.21.25): icmp_seq=37 ttl=64 time=725 ms
64 bytes from z5 (10.150.21.25): icmp_seq=38 ttl=64 time=0.250 ms
64 bytes from z5 (10.150.21.25): icmp_seq=39 ttl=64 time=0.298 ms
64 bytes from z5 (10.150.21.25): icmp_seq=40 ttl=64 time=0.257 ms
^C
--- z5 ping statistics ---
40 packets transmitted, 8 received, +9 errors, 80% packet loss, time 40551ms
rtt min/avg/max/mdev = 0.153/662.097/2804.876/1000.078 ms, pipe 3
arter97@arter97-dgist:~$
```

▲ Regular reboot takes 36s

Limitations

- efifb may not work
  - Display output may not work until the new kernel loads the graphics drivers
- BIOS implementations may break kexec
  - Especially on AMD platforms, updating your BIOS may help
- Some hardware changes may still require traditional reboots
  - e.g., Newly connected HDDs/SSDs may not appear on kexec reboots
- So, test kexec when you have physical access (or IPMI) before deployment!

Extra

- qboot is also extremely helpful for virtualized kernel development
  - New login shell appears within 3s under QEMU
  - https://github.com/bonzini/qboot
  - https://twitter.com/arter97/status/1295671273784147969
- kexec in action
  - https://youtu.be/60Gh1NnK0MA

Thank you!