kdump: usage and internals

CFP, #LinuxCon, Beijing, June 19-20, 2017

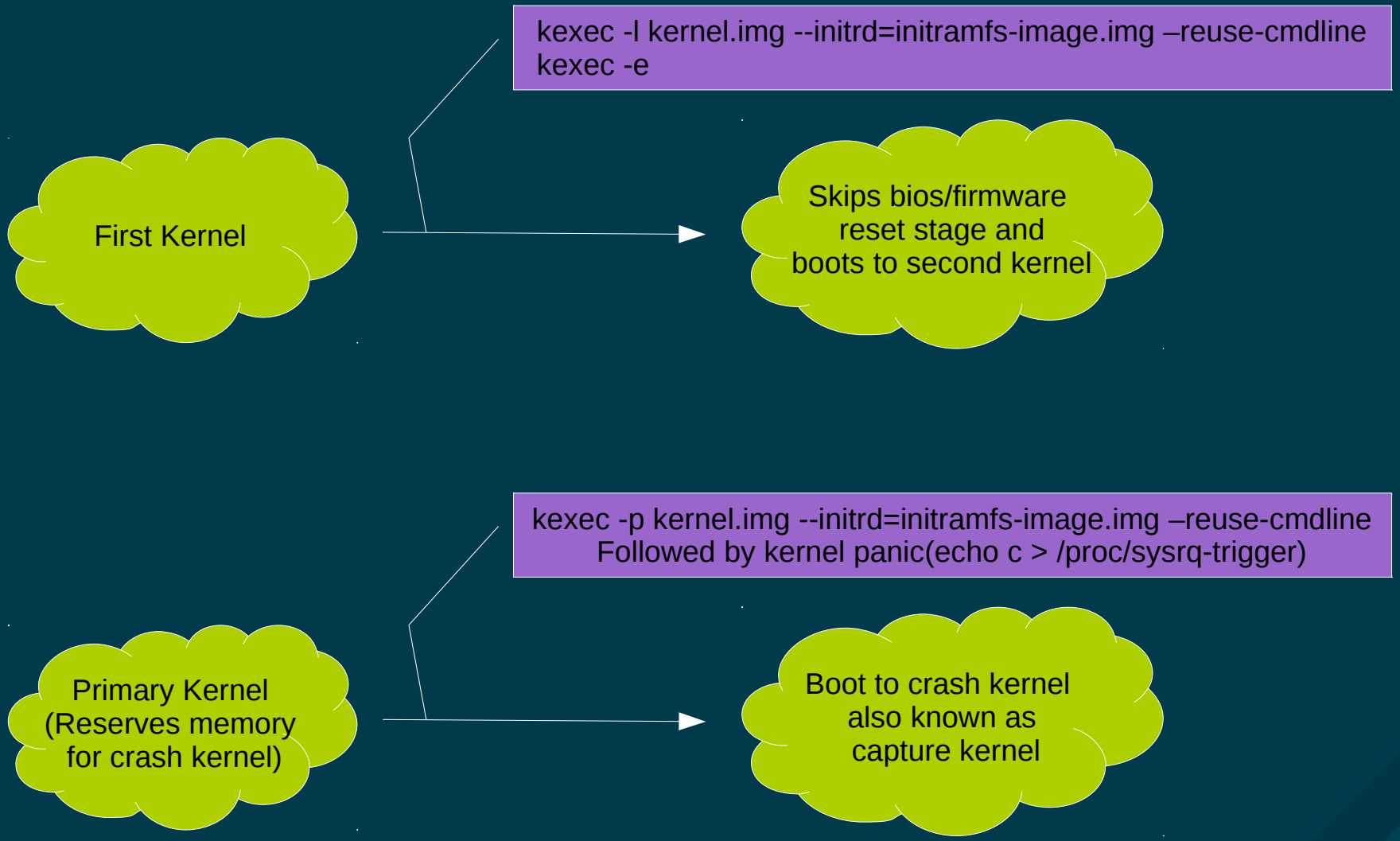
Pratyush Anand(panand@redhat.com)

Agenda

- kdump from user perspective
- Kernel system calls
- When Kernel crashes…
- vmcore structure
- makedumpfile
- kdump: The Fedora way
- Debugging kdump issues
- What next

kdump from user perspective

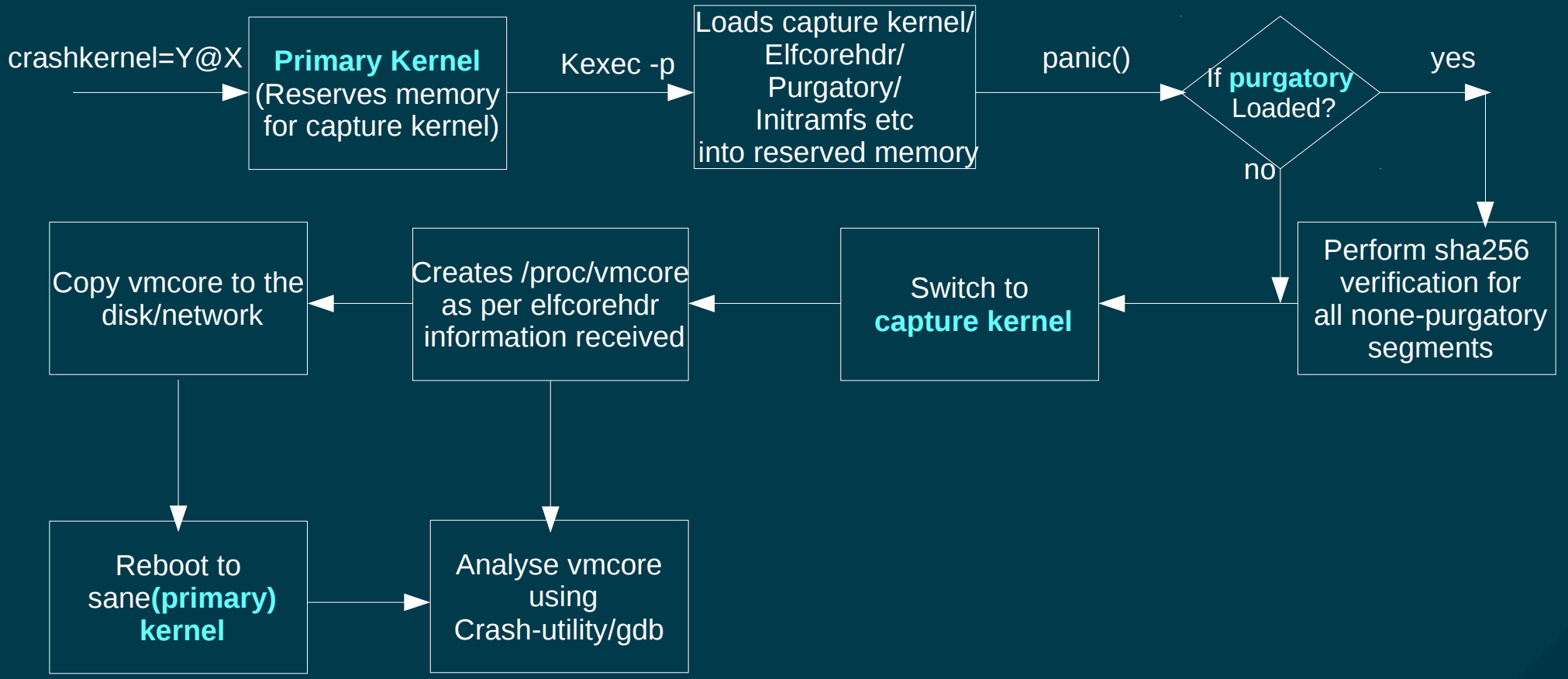
Overview



Different pieces

Linux Kernel (git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git) kernel/kexec* arch/…/kernel/machine_kexec*.c Mailing list: kexec@lists.infradead.org	kexec_load(), kexec_file_load() and reboot() system call Arch specific code like machine_kexec() and machine_crash_shutdown() etc
kexec-tools (git://git.kernel.org/pub/scm/utils/kernel/kexec/kexec-tools.git) Mailing list: kexec@lists.infradead.org	Uses kernel system calls and provides a user command `kexec`
Distribution code (like fedora kexec-tools) (git://pkgs.fedoraproject.org/kexec-tools) Mailing list: kexec@lists.fedoraproject.org	Specs file and scripts to provide user friendly command/services so that kexec-tools can be automated in different user scenarios
makedumpfile (git://git.code.sf.net/p/makedumpfile/code) Mailing list: kexec@lists.infradead.org	Used to compress vmcore and erase sensitive information from it

Kdump : End to end Flow



Demo

https://www.youtube.com/watch?v=iOq_rJhrKhA

Kernel system calls

Kernel: kexec_load() (1)

- The kexec_load() system call loads a new kernel that can be executed later by reboot()
  - long kexec_load(unsigned long entry, unsigned long nr_segments, struct kexec_segment *segments, unsigned long flags);
- User space need to pass segment for different components like kernel, initramfs etc.
  - struct kexec_segment {

```
        void   *buf;        /* Buffer in user space */
        size_t  bufsz;      /* Buffer length in user space */
        void   *mem;        /* Physical address of kernel */
        size_t  memsz;      /* Physical address length */
    };
```

Kernel: kexec_load() (2)

- reboot(LINUX_REBOOT_CMD_KEXEC);
- kexec_load() and above reboot() option is only available when kernel was configured with CONFIG_KEXEC.
- Supported architecture:
  - X86, X86_64, ppc64, ia64, S390x, arm, arm64
- **KEXEC_ON_CRASH**
  - A flag which can be passed to kexec_load()
  - Execute the new kernel automatically on a system crash.
  - CONFIG_CRASH_DUMP should be configured

Kernel: kexec_file_load()

- CONFIG_KEXEC_FILE should be enabled to use this system call.
- It is an in-kernel way of segment preparation.
  - long kexec_file_load(int kernel_fd, int initrd_fd, unsigned long cmdline_len, const char __user * cmdline_ptr, unsigned long flags);
- User space need to pass kernel and initramfs file descriptor.
- Only supported for x86 and powerpc

When Kernel crashes...

When Kernel crashes…..

- Update vmcoreinfo note (crash_save_vmcoreinfo())
- shutdown non-crashing cpus and save registers (machine_crash_shutdown() & crash_save_cpu())
- Might need to disable interrupt controller here
- Perform kexec reboot now (machine_kexec())
  - Load/flush kexec segments to memory
  - Pass control to the execution of entry segment

vmcore structure

Elf Program Headers

- Most of the dump cores involved in kdump are in ELF format.
- Each elf file has a program header
  - Which is read by the system loader
  - Which describes how the program should be loaded into memory.
  - `Objdump -p elf_file` can be used to look into program headers

Elf Program Headers

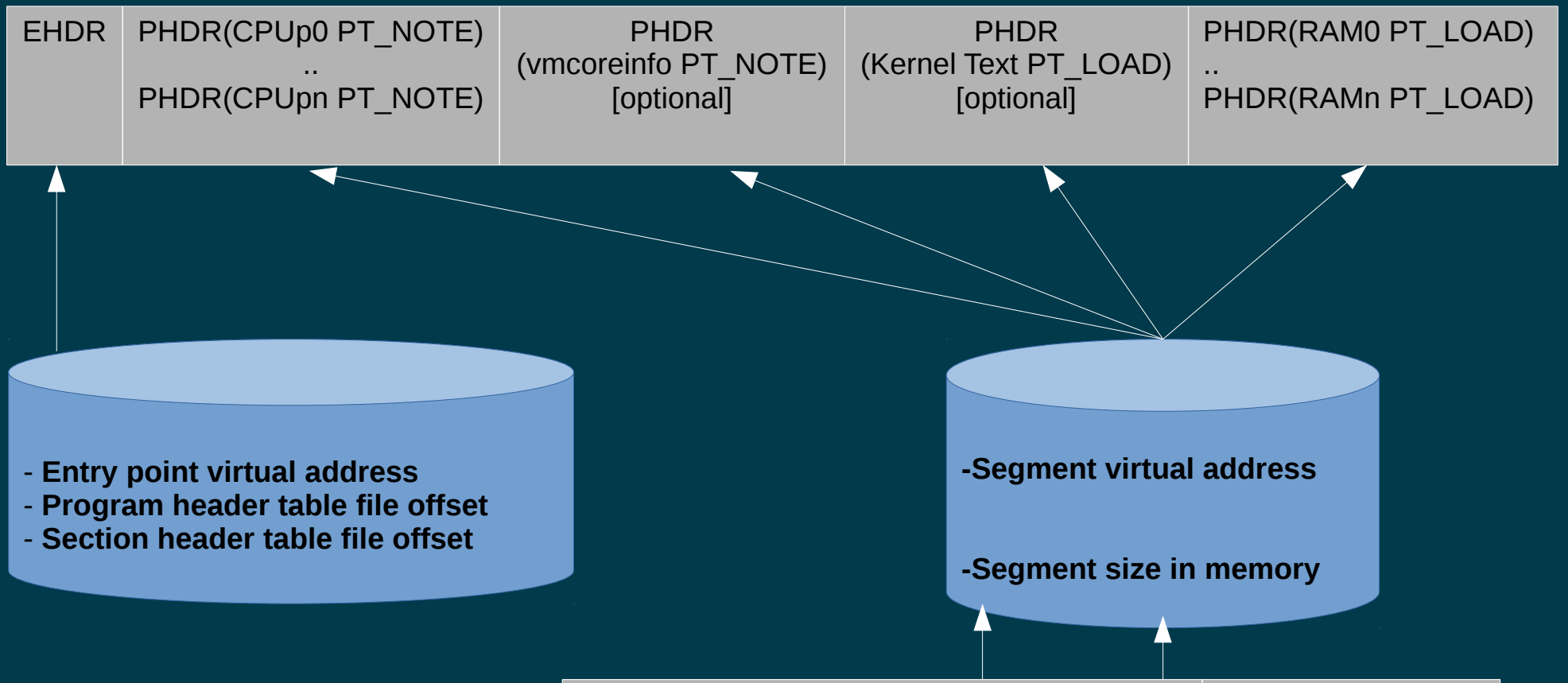
```
# objdump -p vmcore

vmcore:     file format elf64-littleaarch64

Program Header:
    NOTE off    0x0000000000010000 vaddr 0x0000000000000000 paddr 0x0000000000000000 align 2**0 filesz
0x00000000000013e8 memsz 0x00000000000013e8 flags ---
    LOAD off    0x0000000000020000 vaddr 0xffff000008080000 paddr 0x0000004000280000 align 2**0 filesz
0x0000000001460000 memsz 0x0000000001460000 flags rwx
    LOAD off    0x0000000001480000 vaddr 0xffff800000200000 paddr 0x0000004000200000 align 2**0 filesz
0x000000007fc00000 memsz 0x000000007fc00000 flags rwx
    LOAD off    0x0000000081080000 vaddr 0xffff8000ffe00000 paddr 0x00000040ffe00000 align 2**0 filesz
0x00000002fa7a0000 memsz 0x00000002fa7a0000 flags rwx
    LOAD off    0x000000037b820000 vaddr 0xffff8003fa9e0000 paddr 0x00000043fa9e0000 align 2**0 filesz
0x0000000004fc0000 memsz 0x0000000004fc0000 flags rwx
    LOAD off    0x00000003807e0000 vaddr 0xffff8003ff9b0000 paddr 0x00000043ff9b0000 align 2**0 filesz
0x0000000000010000 memsz 0x0000000000010000 flags rwx
    LOAD off    0x00000003807f0000 vaddr 0xffff8003ff9f0000 paddr 0x00000043ff9f0000 align 2**0 filesz
0x0000000000610000 memsz 0x0000000000610000 flags rwx

private flags = 0:
```

elfcorehdr



/sys/devices/system/cpu/cpu%d/crash_notes	CPU PT_NOTE
/sys/kernel/vmcoreinfo	vmcoreinfo PT_NOTE
/proc/iomem	Mem PT_LOAD

Notes

- Crash notes
  - A percpu area for storing cpu states in case of system crash
  - Has information about current pid and cpu registers
- Vmcoreinfo
  - This note section has various kernel debug information like struct size, symbol values, page size etc.
  - Vmcoreinfo is used mainly by makedumpfile application
  - include/linux/kexec.h has macros to define a new vmcoreinfo
    - VMCOREINFO_PAGESIZE()
    - VMCOREINFO_SYMBOL()
    - VMCOREINFO_SIZE()

makedumpfile

makedumpfile

- It compresses /proc/vmcore data
- Excludes unnecessary pages like:
  - Pages filled with zero
  - Cache pages without private flag (non-private cache)
  - Cache pages with private flag (private cache)
  - User process data pages
  - Free pages
- Needs first kernel’s debug information to exclude unnecessary pages

makedumpfile

- Debug information comes from either VMLINUX or VMCOREINFO
- Can also erase any specific sensitive kernel symbol
- Output can either be in ELF format or kdump-compressed format
- Typical usage:
  - makedumpfile -l --message-level 1 -d 31 /proc/vmcore makedumpfilecore
    - -d to specify the type of unnecessary page for analysis.
    - -l compression using lzo
    - -c compression using zlib
    - -p compression using snappy

Kdump: The Fedora way

Pratyush Anand

Kdump: The Fedora way

- Start/stop/status kdump service:
  - systemctl start kdump
  - systemctl stop kdump
  - systemctl status kdump
- Fedora has some scripts to take care of various use case scenarios.
  - Kdumpctl
  - mkdumprd
- Configurations files:
  - /etc/sysconfig/kdump:
    - Initrd rebuild is not needed after any configuration change
  - /etc/kdump.conf:
    - Values which can affect initrd rebuild

Debugging Kdump issues

Debugging Kdump issues

- `Kexec -p kernel_image` did not succeed
  - Check if crash memory is allocated
    - cat /sys/kernel/kexec_crash_size
      - Should have none zero value
    - If not allocated, then pass proper “crashkernel=” argument in command line
    - If nothing shows up then pass -d in the kexec command and share debug output with kexec mailing list.

Debugging Kdump issues

- Do not see anything on console after last message from first kernel (like “bye”):
  - Might have second kernel crashed very early
    - Pass some earlycon/earlyprintk option for your system to the second kernel command line
    - Share dmesg log of both 1st and 2nd kernel with kexec mailing list.
  - Check if `kexec -l kernel_image’ followed by `kexec -e` works
  - Might be missing some arch/machine specific options
  - Might have purgatory sha verification failed. If your arch does not support a console in purgatory then it is very difficult to debug.

What Next

What next

- kexec_file_load() support for unsupported arch
- shrink memory use for kdump initramfs
- move distribution initramfs code to upstream
- simplify kdump setup

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos