How to Compile and Install Linux Kernel v4.5 Source On a Debian / Ubuntu Linux

by Vivek Gite on September 11, 2015 *last updated* March 21, 2016
in Debian / Ubuntu, Linux, Package Management, Ubuntu Linux

How do I download, compile and install the latest version of the Linux kernel on a Debian Linux v8.x or Ubuntu Linux LTS home server or my laptop? How do I build and install a custom Linux kernel on a Debian or Ubuntu Linux based system?

In order to create a custom kernel configuration file and build a custom kernel, the full Linux kernel source tree must first be downloaded and installed. The latest Linux kernel stable version is **4.5**. In this tutorial, you will **learn how to compile the Linux kernel version 4.5 on a Debian and Ubuntu Linux** operating system and build .deb file.

Why build a custom kernel?

Compiling a custom Linux kernel has its advantages and disadvantages. To change the kernel's behavior, one had to compile and then reboot into a new Linux. Most of the functionality in the Linux kernel contained in modules that can be dynamically loaded and unloaded from the kernel as necessary. Some benefits of a custom Linux kernel:

1. Support a wide range of hardware including the latest hardware.
2. Remove unwanted drivers from the kernel.
3. Faster boot time due to small kernel size.
4. Increased security due to additional or removed modules/drivers/features.
5. You will learn about the kernel and advanced usage.
6. Always run the cutting edge latest kernel.
7. Lower memory usage.

Note: The following instructions were tested on both Debian Linux v8.x and Ubuntu Linux v14.04.4 LTS.

Prerequisites

You need to install the following packages on a Debian or Ubuntu Linux to compiler the Linux kernel:

- **git** : Fast, scalable, distributed revision control system. You can grab the latest source code using the git command.
- **fakeroot** : Tool for simulating superuser privileges. Useful to build .deb files.

- **build-essential** : Tools for building the Linux kernel such as GCC compiler and related tools on a Debian or Ubuntu Linux based system.
- **ncurses-dev** : Developer's libraries for ncurses. This is used by menuconfig while configuring the kernel options.
- **kernel-package** : Utility for building Linux kernel related Debian packages.
- **xz-utils** : XZ-format compression utilities to decompress the Linux kernel tar ball.
- **Disk space** : 10 GB or more free disk space.
- **Time** : Kernel compilation may take quite a while, depending on the power of your machine.

Install required packages

Open the terminal application. Type the following apt-get command to install the required packages for building the Linux kernel:

```
$ sudo apt-get install git fakeroot build-essential ncurses-dev
xz-utils libssl-dev bc
```

Sample outputs:

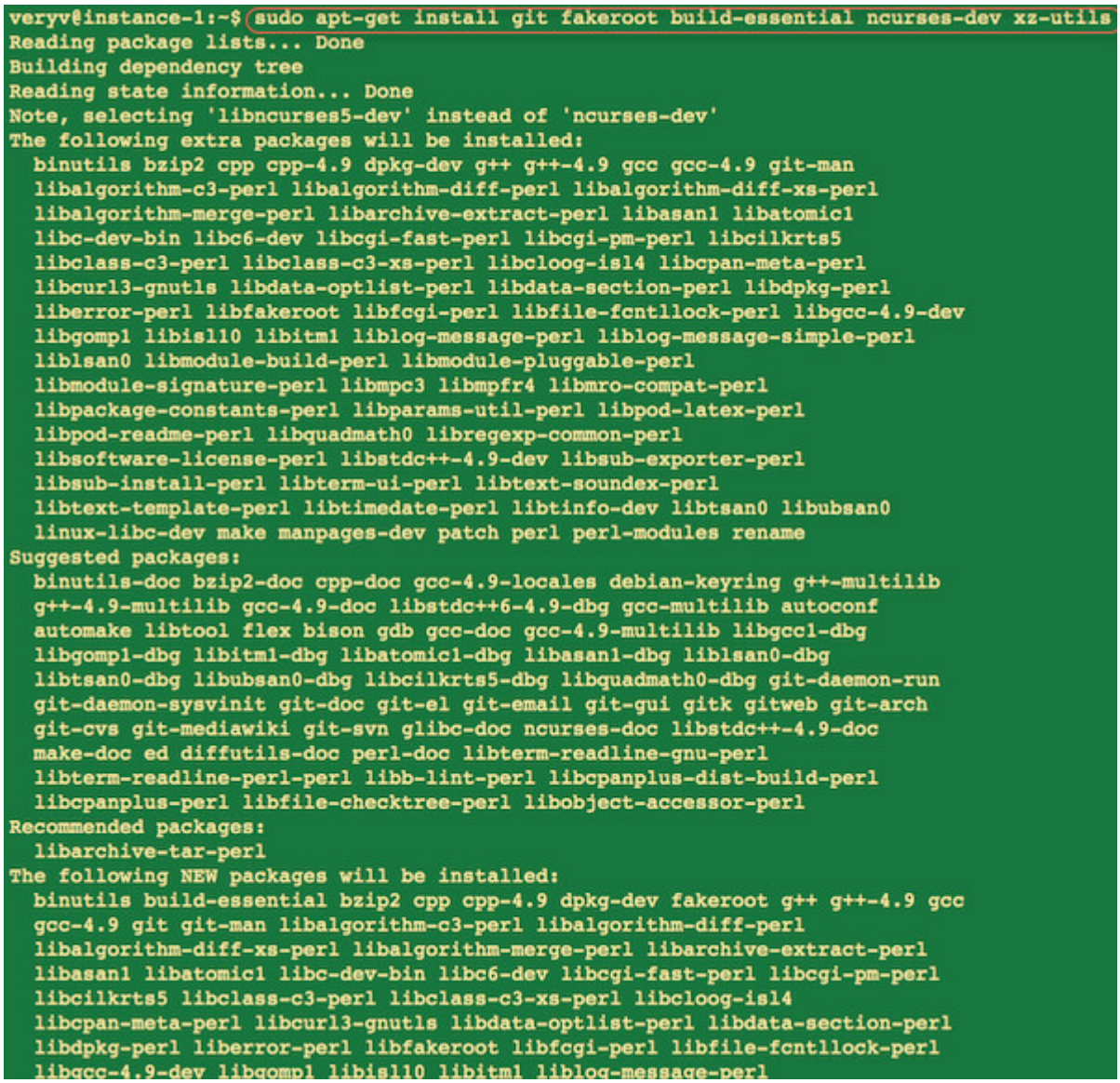

Fig.01: Install gcc and friends

Finally, install the kernel-package package too:

```
$ sudo apt-get install kernel-package
```

OR

```
$ sudo apt-get --no-install-recommends install kernel-package
```

Sample outputs:

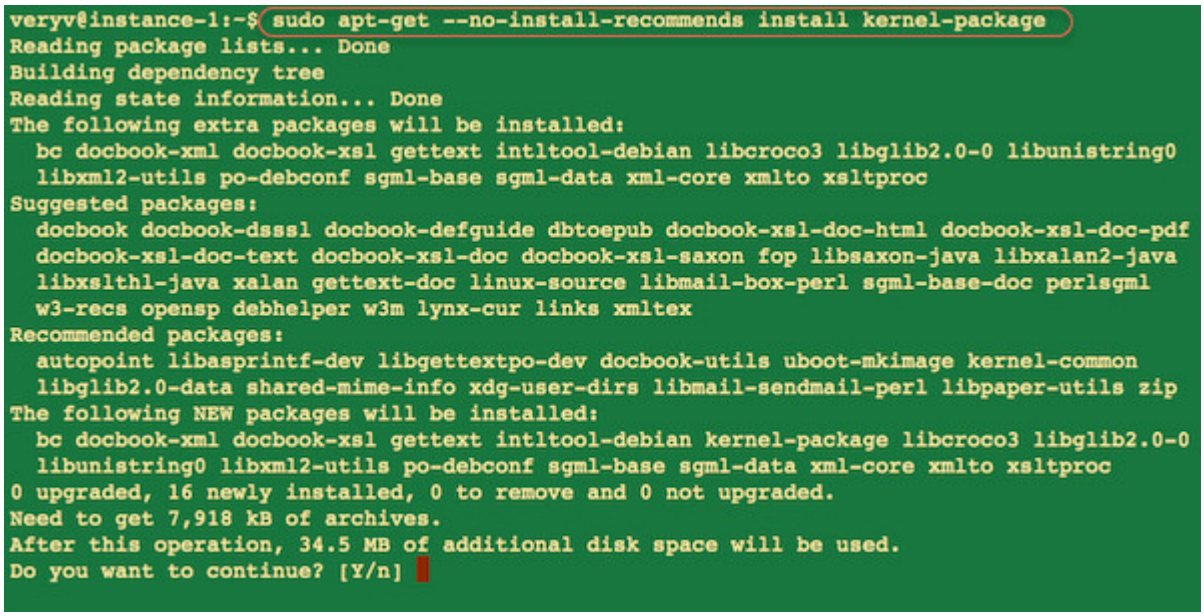

Fig.02: Install utility for building Linux kernel

Download the Linux kernel source code

Type the following wget command:

```
$ wget https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.5.tar.xz
```

Sample outputs:

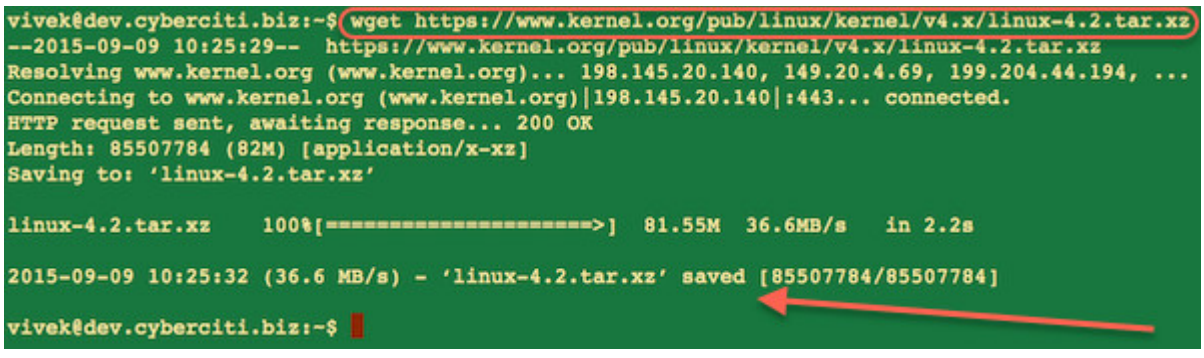

Fig.03: Use the wget to grab the latest source code from kernel.org

Use GnuPG to verify kernel signatures:

```
$ unxz linux-4.5.tar.xz
```

```
$ gpg --verify linux-4.5.tar.sign
```

Sample outputs:

```
gpg: assuming signed data in `linux-4.5.tar'
gpg: Signature made Mon 14 Mar 2016 04:35:48 AM UTC using RSA key ID 00411886
gpg: Can't check signature: public key not found
```

Get the public key from the PGP keyserver in order to verify the signature i.e. RSA key ID **00411886** (from the above outputs):

```
$ gpg --keyserver hkp://keys.gnupg.net --recv-keys 00411886
```

Sample outputs:

```
gpg: requesting key 00411886 from hkp server keys.gnupg.net
gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: key 00411886: public key "Linus Torvalds " imported
gpg: no ultimately trusted keys found
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

Now verify again:

```
$ gpg --verify linux-4.5.tar.sign
```

Sample outputs:

```
gpg: assuming signed data in `linux-4.5.tar'
gpg: Signature made Mon 14 Mar 2016 04:35:48 AM UTC using RSA key ID 00411886
gpg: Good signature from "Linus Torvalds "
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: ABAF 11C6 5A29 70B1 30AB  E3C4 79BE 3E43 0041 1886
```

If you do not "**BAD signature**" output from "`gpg --verify`" command, untar the Linux kernel tar ball using the tar command enter:

```
$ tar xvf linux-4.5.tar
$ ls
$ cd linux-4.5
$ ls
```

Sample outputs

```
linux-4.5/
linux-4.5/.get_maintainer.ignore
linux-4.5/.gitignore
linux-4.5/.mailmap
linux-4.5/COPYING
linux-4.5/CREDITS
linux-4.5/Documentation/
linux-4.5/Documentation/00-INDEX
linux-4.5/Documentation/ABI/
linux-4.5/Documentation/ABI/README
linux-4.5/Documentation/ABI/obsolete/
linux-4.5/Documentation/ABI/obsolete/proc-sys-vm-nr_pdflush_threads
....
...
linux-4.5/virt/kvm/async_pf.h
linux-4.5/virt/kvm/coalesced_mmio.c
linux-4.5/virt/kvm/coalesced_mmio.h
linux-4.5/virt/kvm/eventfd.c
linux-4.5/virt/kvm/irqchip.c
```

```
linux-4.5/virt/kvm/kvm_main.c
linux-4.5/virt/kvm/vfio.c
linux-4.5/virt/kvm/vfio.h
linux-4.5/virt/lib/
linux-4.5/virt/lib/Kconfig
linux-4.5/virt/lib/Makefile
linux-4.5/virt/lib/irqbypass.c
linux-4.5
```

```
arch     crypto          include  kernel       net              security
block    Documentation   init     lib          README           sound
certs    drivers         ipc      MAINTAINERS  REPORTING-BUGS   tools
COPYING  firmware        Kbuild   Makefile     samples          usr
CREDITS  fs              Kconfig  mm           scripts          virt
```

Configure the Linux kernel

First, copy your existing Linux kernel config file

```
$ cd linux-4.5
$ cp /boot/config-$(uname -r) .config
```

To configure the kernel, run:

```
$ make menuconfig
```

Sample outputs:

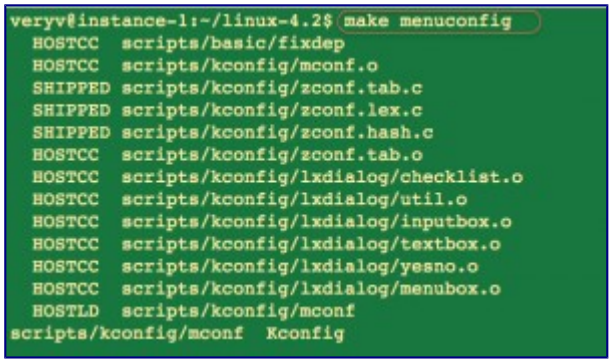

Fig.04: Starting menuconfig

Fig.05: Select Linux kernel config options and drivers to build

Gallery 01: Click to enlarge

WARNING: It is easy to remove support for a device driver or option and end up with a broken kernel. For example, if the ext4 driver is removed from the kernel configuration file, a system may not boot. When in doubt, just leave support in the kernel.

Make sure you save the changes before exit from menuconfig.

Compile the Linux kernel

You need to clean the source tree and reset the kernel-package parameters, type:

```
$ make-kpkg clean
```

Sample outputs:

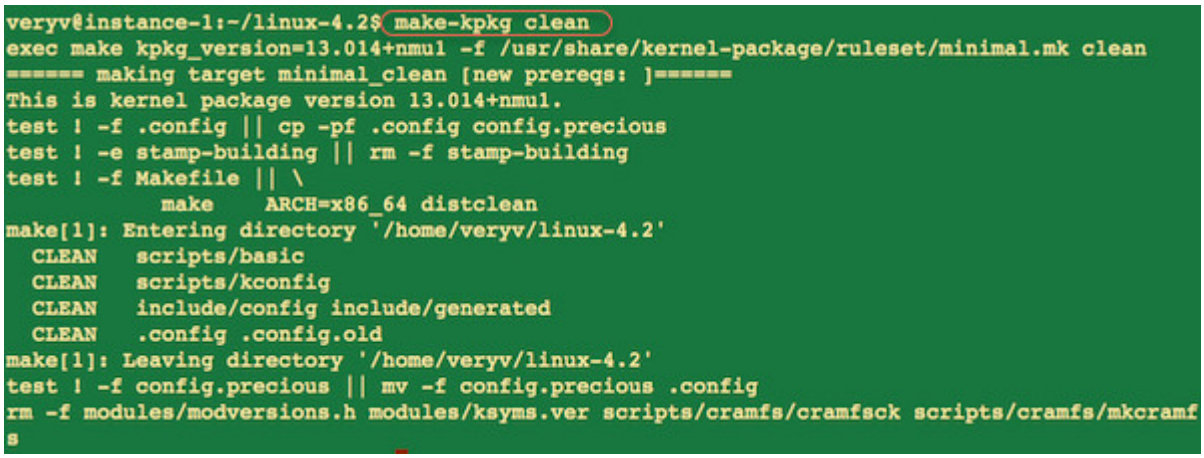

Fig.06: Run make-kpkg command

Now, you can compile the kernel, run:

```
$ fakeroot make-kpkg --initrd --revision=1.0.NAS kernel_image
kernel_headers
```

To speed up the compile process pass the `-j` option (-j 16 means you are using all 16 cores to compile the Linux kernel):

```
$ fakeroot make-kpkg --initrd --revision=1.0.NAS kernel_image
kernel_headers -j 16
```

Sample outputs:

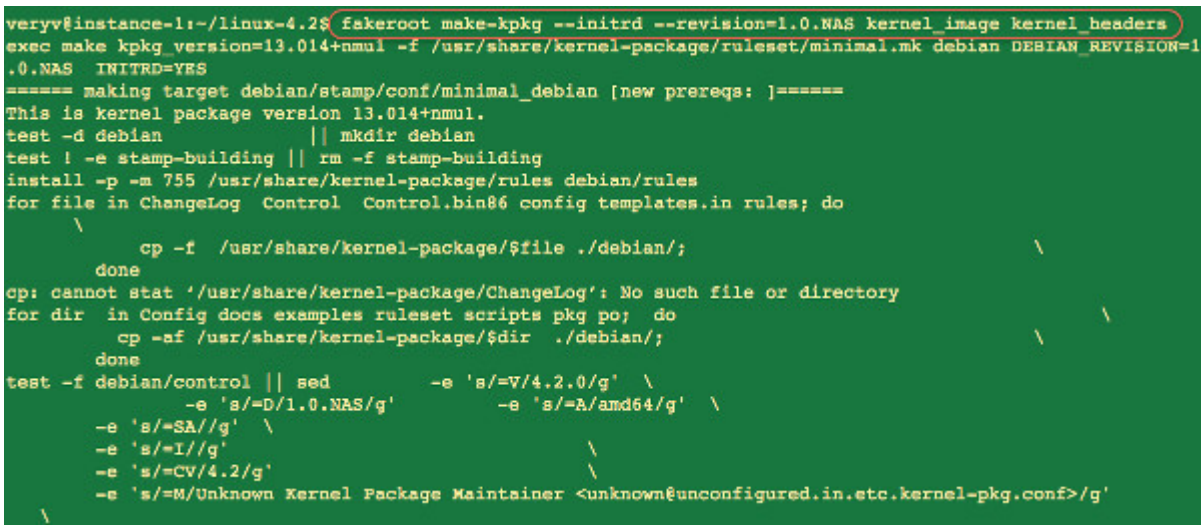

Fig.07: Start compiling the kernel

The fakeroot runs a command called make-kpkg in an environment wherein it appears to have root privileges for file manipulation. This is useful for allowing users to create archives (tar, ar, .deb etc.) with files in them with root permissions/ownership. The make-kpkg command build Debian/Ubuntu kernel packages from Linux kernel sources and options are:

- `--initrd` : Create an initrd image.
- `--revision=1.0.NAS` : Set custom revision for your kernel such as 1.0.NAS or -1.0-custom-kernel etc.
- `kernel_image` : This target produces a Debian package of the Linux kernel source image, and any modules configured in the kernel configuration file .config.
- `kernel_headers` : This target produces a Debian package of the Linux kernel header image.

Please note that kernel compilation may take quite a while, depending on the power of your machine. On my shared 4 CORE CPU and 4GB ram it took 60 mins to build the Linux kernel. In the end you should see something as follows on screen:

```
test ! -e debian/control~ || rm -f debian/control~
dpkg-gencontrol -isp -DArchitecture=amd64 -plinux-headers-4.5.0 \
                                          -P/root/linux-4.5/debian/linux-
headers-4.5.0/
dpkg-gencontrol: warning: -isp is deprecated; it is without effect
create_md5sums_fn () { cd $1 ; find . -type f ! -regex './DEBIAN/.*' ! -regex
'./var/.*'      -printf '%P\0' | xargs -r0 md5sum > DEBIAN/md5sums ; if [ -z
"DEBIAN/md5sums" ] ; then rm -f "DEBIAN/md5sums" ; fi ; } ; create_md5sums_fn
/root/linux-4.5/debian/linux-headers-4.5.0
chown -R root:root                  /root/linux-4.5/debian/linux-headers-4.5.0
chmod -R og=rX                      /root/linux-4.5/debian/linux-headers-4.5.0
dpkg --build                        /root/linux-4.5/debian/linux-headers-
4.5.0 ..
dpkg-deb: building package `linux-headers-4.5.0' in `../linux-headers-
4.5.0_1.0.NAS_amd64.deb'.
cp -pf debian/control.dist          debian/control
make[2]: Leaving directory '/root/linux-4.5'
make[1]: Leaving directory '/root/linux-4.5'
```

Verify kernel deb files:

```
$ ls  ../*.deb
../linux-headers-4.5.0_1.0.NAS_amd64.deb  ../linux-image-4.5.0_1.0.NAS_amd64.deb
```

Installing a custom kernel

Type the following dpkg command to install a custom kernel on your system:

```
$ sudo dpkg -i linux-headers-4.5.0_1.0.NAS_amd64.deb
$ sudo dpkg -i linux-image-4.5.0_1.0.NAS_amd64.deb
```

Sample outputs:

```
Selecting previously unselected package linux-image-4.5.0.
(Reading database ... 251501 files and directories currently installed.)
Preparing to unpack linux-image-4.5.0_1.0.NAS_amd64.deb ...
Examining /etc/kernel/preinst.d/
Done.
Unpacking linux-image-4.5.0 (1.0.NAS) ...
Setting up linux-image-4.5.0 (1.0.NAS) ...

 Hmm. There is a symbolic link /lib/modules/4.5.0/build
 However, I can not read it: No such file or directory
 Therefore, I am deleting /lib/modules/4.5.0/build


 Hmm. The package shipped with a symbolic link /lib/modules/4.5.0/source
 However, I can not read the target: No such file or directory
 Therefore, I am deleting /lib/modules/4.5.0/source

Running depmod.
Examining /etc/kernel/postinst.d.
run-parts: executing /etc/kernel/postinst.d/apt-auto-removal 4.5.0 /boot/vmlinuz-4.5.0
run-parts: executing /etc/kernel/postinst.d/dkms 4.5.0 /boot/vmlinuz-4.5.0
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 4.5.0 /boot/vmlinuz-4.5.0
update-initramfs: Generating /boot/initrd.img-4.5.0
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 4.5.0 /boot/vmlinuz-4.5.0
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-4.5.0
Found initrd image: /boot/initrd.img-4.5.0
Found linux image: /boot/vmlinuz-4.4.4
Found initrd image: /boot/initrd.img-4.4.4
Found linux image: /boot/vmlinuz-4.4.2
Found initrd image: /boot/initrd.img-4.4.2
Found linux image: /boot/vmlinuz-4.4.0
Found initrd image: /boot/initrd.img-4.4.0
Found linux image: /boot/vmlinuz-4.3.0
Found initrd image: /boot/initrd.img-4.3.0
Found linux image: /boot/vmlinuz-4.2.5
Found initrd image: /boot/initrd.img-4.2.5
Found linux image: /boot/vmlinuz-4.2.0
Found initrd image: /boot/initrd.img-4.2.0
done
```

Reboot the box/server/laptop

Type the following command:

```
$ reboot
```

OR

```
$ shutdown -r now
```

Verify that everything is working

Type the following command to verify your new kernel and everything is working fine:

```
$ uname -a
$ uname -r
$ uname -mrs
$ dmesg | more
$ dmesg | egrep -i --color 'error|critical|failed'
```

Sample outputs:

```
Linux nas02.nixcraft.net.in 4.5.0 #1 SMP Mon Mar 21 05:25:25 UTC 2016 x86_64
GNU/Linux
```

And, there you have it, the Linux kernel version 4.4.4 installed and working correctly.