➔ How to build the kernel from source:

For experts only:

You can remove the need to prefix commands with sudo by starting a shell with sudo su –

Warning: the – character does not copy and paste correctly between a pdf and text file.

1. Go to kernel.org and download source for building a version of the kernel:

   https://www.kernel.org/pub/linux/kernel/v4.x/

   Choose the long term support version closest to the current working version on your system. (uname –r)

2. Also grab the associated pgp file for your kernel.
3. As the super-user, un-tar the compressed file in directory in /usr/src :
   - cd /usr/src
   - sudo tar xvf /path/tarfile.tar

4. Navigate to the new directory and check for a .config file. If it exists move it old.config. Confirm that the source is extracted from the tar file. There may not be a .config file with the tarball.

   - sudo mv .config old.config (note the file may not exist)

5. Prepare for the build:
   - cd /usr/src/linux.x.xx.x (current installed version)
   - sudo make clean
   - sudoudo make mrproper

6. Identify the current kernel and copy the current working config-file (uname –r) to the new directory with the source.

   - sudo cp /boot/config-4.xx.x-xx-generic /usr/src/linux-4.xx.x.xx
   - cd /usr/src/linux-*newVersion*
   - sudo cp config-4.xx.x-xx-generic .config

7. Install the curses interface for modifying the .config file:
   - sudo apt-get install libncurses5-dev

8. sudo make menuconfig This will allow you to modify the .config file!

   Find something to remove: Use the arrow keys to navigate and the spacebar to modify entries.

   Save and Exit your changes. Note: the interface has a minimum window size, if there is an error, try resizing the terminal-window.

9. Confirm the new .config is different from the original.

- sudo apt-get install libssl-dev <- necessary for build, you need gcc too.

10. Build the kernel using the make utility:

    - sudo make –j2 <- This takes about 55-min using a lab computer.
    - sudo make modules_install

      look at /lib/modules/versionNumber/ (uname –r for version number)

    sudo make install

11. Confirm the new kernel installation by checking the content of /boot: ls –l /boot.

12. Before you try the new kernel, set up the boot-loader (grub) so it will default-boot the last known working kernel!

    a. Get the current kernel version: uname –a
    b. Open for read only: view /boot/grub/grub.cfg
       And copy elementary, with Linux x.xx.x-xx-generic
       Where x.xx.x-xx is the current working kernel (uname –r)
    c. Edit /etc/default/grub and paste so that:

    Example:

    GRUB_DEFAULT= 'gnulinux-advanced-70a92550-52e1-4362-a77d-badae821f60d>gnulinux-4.13.0-32-generic-advanced-70a92550-52e1-4362-a77d-badae821f60d'

    Comment out the line:

    #GRUB_HIDDEN_TIMEOUT=0

    - sudo grub-mkconfig -o /boot/grub/grub.cfg
    - sudo update-grub

    reboot

13. If there are problems. You can boot in single user mode as root, or without the GUI. If these work, then your build was partially successful.

14. If everything works you need to do some cleanup, navigate to the /usr/src/linux-xxx where the new source is located:

- df
- sudo make clean
- df
- sudo make mrproper
- df

15. To remove the new kernel go to /usr/src/newVersion and delete everything in that directory.

- cd /usr/src/newVersion
- sudo rm –rf *
- cd /usr/src
- sudo rmdir newVersion (Optional; and, you may need to remove some hidden files first)
- cd /boot/
- sudo rm all the new version files only! Be careful removing files here.
- cd /etc/default
- sudo grub-mkconfig -o /boot/grub/grub.cfg
- sudo update-grub

Confirm the new kernel is removed from the /etc/default grub and /boot/grub/grub.cfg files.

reboot

Appendix

If there is something that goes wrong:

sudo apt-get remove linux-generic-lts-raring

Which will backout the changes.

Appendix

https://www.youtube.com/watch?v=SwbRpGW2Qyw