# How to build an Ubuntu Linux kernel

If you have patches you need to apply to the Ubuntu Linux kernel, or you want to change some kernel configs, you may need to build your kernel from source. Follow these steps to customise and build the Ubuntu Linux kernel locally.

## Prerequisites

* This guide supports Xenial Xerus and newer.
* It is recommended to have at least 8GB of RAM and 30GB free disk space on the build machine.

If this is the first time you are building a kernel on your system, you will need to [Set up build environment](#how-to-build-kernel-setup) and [Install required packages](#how-to-build-kernel-install-packages).

Otherwise, skip ahead to [Obtain the source for an Ubuntu release](#how-to-build-kernel-obtain-source).

### Set up build environment

To build an Ubuntu kernel, you will need to enable the necessary source repositories in the sources.list or ubuntu.sources file.

* Noble Numbat 24.04 (and newer)
* Add “deb-src” to the Types: line in the /etc/apt/sources.list.d/ubuntu.sources file.
* Types: deb deb-src  
  URIs: http://archive.ubuntu.com/ubuntu  
  Suites: noble noble-updates noble-backports  
  Components: main universe restricted multiverse  
  Signed-By: /usr/share/keyrings/ubuntu-archive-keyring.gpg
* Mantic Minotaur 23.10 (and older)
* Check that you have the following entries in the /etc/apt/sources.list file. If not, add or uncomment these lines for your Ubuntu release.
* deb-src http://archive.ubuntu.com/ubuntu jammy main  
  deb-src http://archive.ubuntu.com/ubuntu jammy-updates main

### Install required packages

To install the required packages and build dependencies, run:

sudo apt update  
sudo apt build-dep -y linux linux-image-unsigned-$(uname -r)  
sudo apt install -y fakeroot llvm libncurses-dev dwarves

## Obtain the source for an Ubuntu release

There are different ways to get the kernel sources, depending on the kernel version you want to make changes to.

### Get kernel source for version installed on build machine

Use the apt source command to get the source code for the kernel version currently running on your build machine.

apt source linux-image-unsigned-$(uname -r)

This will download and unpack the kernel source files to your current working directory.

<working\_directory>  
├── linux-X.Y.Z/  
│ └── \*  
├── linux\_X.Y.Z-\*.diff.gz  
├── linux\_X.Y.Z-\*.dsc  
└── linux\_X.Y.Z.orig.tar.gz

### Get kernel source for other versions

Use Git to get the source code for other kernel versions. See [How to obtain kernel source for an Ubuntu release using Git](../../prepare/obtain-kernel-source-git/) for detailed instructions.

## Prepare the kernel source

Once you have the kernel source, go to the kernel source working directory (e.g. “linux-6.8.0”) and run the following commands to ensure you have a clean build environment and the necessary scripts have execute permissions:

cd <kernel\_source\_working\_directory>  
chmod a+x debian/rules  
chmod a+x debian/scripts/\*  
chmod a+x debian/scripts/misc/\*  
fakeroot debian/rules clean

### Modify ABI number

You should modify the kernel version number to avoid conflicts and to differentiate the development kernel from the kernel released by Canonical.

To do so, modify the ABI number (the number after the dash following the kernel version) to “999” in the first line of the <kernel\_source\_working\_directory>/debian.master/changelog file.

For example, modify the ABI number to “999” for Noble Numbat:

linux (6.8.0-999.48) noble; urgency=medium

If you are building something other than the generic Ubuntu Linux kernel, modify the ABI number in the <kernel\_source\_working\_directory>/debian.<derivative>/changelog file instead.

## Modify kernel configuration

(Optional) To enable or disable any features using the kernel configuration, run:

cd <kernel\_source\_working\_directory>  
fakeroot debian/rules editconfigs

This will invoke the menuconfig interface for you to edit specific configuration files related to the Ubuntu kernel package. You will need to explicitly respond with Y or N when making any config changes to avoid getting errors later in the build process.

## Customise the kernel

(Optional) Add any firmware, binary blobs, or patches as needed.

## Build the kernel

You are now ready to build the kernel.

cd <kernel\_source\_working\_directory>  
fakeroot debian/rules clean  
fakeroot debian/rules binary

***Note****: Run fakeroot debian/rules clean to clean the build environment each time before you recompile the kernel after making any changes to the kernel source or configuration.*

If the build is successful, several .deb binary package files will be produced in the directory one level above the kernel source working directory.

For example, building a kernel with version “6.8.0-999.48” on an x86-64 system will produce the following .deb packages (and more):

* linux-headers-6.8.0-999\_6.8.0-999.48\_all.deb
* linux-headers-6.8.0-999-generic\_6.8.0-999.48\_amd64.deb
* linux-image-unsigned-6.8.0-999-generic\_6.8.0-999.48\_amd64.deb
* linux-modules-6.8.0-999-generic\_6.8.0-999.48\_amd64.deb

## Install the new kernel

Install all the debian packages generated from the previous step (on your build system or a different target system with the same architecture) with dpkg and reboot:

cd <kernel\_source\_working\_directory>/../  
sudo dpkg -i linux-headers-<kernel version>\*\_all.deb  
sudo dpkg -i linux-headers-<kernel version>-<generic or derivative>\*.deb  
sudo dpkg -i linux-image-unsigned-<kernel version>-<generic or derivative>\*.deb  
sudo dpkg -i linux-modules-<kernel version>-<generic or derivative>\*.deb  
sudo reboot

## Test the new kernel

Run any necessary testing to confirm that your changes and customisations have taken effect. You should also confirm that the newly installed kernel version matches the value in the <kernel\_source\_working\_directory>/debian.master/changelog file by running:

uname -r