# How to build an Ubuntu Linux kernel snap

If you are running an Ubuntu Core system and want to use boot into a custom kernel, you will need a kernel snap.

This guide shows how to build a kernel snap for local development and testing.

***Important****: Kernel snaps built using this method are not intended for use in production.*

## Prerequisites

Before you begin, you will need:

* A Launchpad account
* To be part of the Launchpad team that owns the project (for private repositories)
* A build machine running Ubuntu
* A device running an Ubuntu Core image with “dangerous” model assertion grade to install the custom kernel snap
* ***Note****: The Ubuntu version of the build host must match the version of the device where the kernel snap will be installed. For example, use an Ubuntu 22.04 (Jammy) host to build the kernel snap for an Ubuntu Core 22 device.*  
  *See* [*Snap - Build environment options*](https://snapcraft.io/docs/build-options#heading--snapcraft) *for more information.*

## Set up build environment

Set up the host machine which will be used to build the kernel snap.

### Install snapcraft

Snapcraft is used to create a managed environment to build the kernel snap. You are recommended to use the latest/stable version of the snapcraft snap from the Snap Store.

On the build machine, remove any existing snapcraft debian package and install snapcraft by running:

sudo apt-get update  
sudo apt-get -y upgrade  
sudo apt purge -y snapcraft  
sudo snap install snapcraft --classic

### Configure source repositories

Configure the package source repositories for the host architecture by specifying the architecture (e.g. “[arch=amd64]” for x86-64 hosts) for each deb source list in the data sources file.

* Ubuntu 24.04 (Noble) and newer
* Update the /etc/apt/sources.list.d/ubuntu.sources file. For example, on a x86-64 host running Ubuntu 24.04 (Noble):
* [...]  
  Types: deb deb-src  
  URIs: http://archive.ubuntu.com/ubuntu  
  Suites: noble noble-updates noble-backports  
  Components: main universe restricted multiverse  
  Architectures: amd64  
  [...]
* Ubuntu 23.10 (Mantic) and older
* Update the /etc/apt/sources.list file. For example, on a x86-64 host running Ubuntu 22.04 (Jammy):
* deb [arch=amd64] http://archive.ubuntu.com/ubuntu focal main restricted
* Alternatively, if you are running a default installation of Ubuntu, you can do a global update of all sources in the /etc/apt/sources.list file.
* sudo sed -ie 's/deb http/deb [arch=amd64] http/g' /etc/apt/sources.list

### Add support for cross-compilation

Add the target architecture (e.g. “arm64”) to the list of supported architectures. This step is only required if the build machine is running on a different architecture than the target device for the kernel snap.

For example, if you want to build a kernel snap for an ARM64 device on a x86-64 host, run:

sudo dpkg --add-architecture arm64  
sudo apt update

Confirm that support for the target architecture has been added successfully by running dpkg –print-foreign-architectures:

user@host:~$ dpkg --print-foreign-architectures

arm64

### Configure SSH settings for Launchpad access

Enable SSH access to git.launchpad.net for your Launchpad account. This step is only required if you are building a snap from a private repository in Launchpad.

Add the following in the ~/.ssh/config file:

Host git.launchpad.net  
 User <your Launchpad username>

## Clone the kernel snap recipe

Once you have set up your host machine, clone the Ubuntu Linux kernel snap recipe.

git clone <kernel-source-repository>

## Customise the kernel

Add any firmware or binary blobs, or customise initrd as needed. This step is only required if you want to make your own changes to the kernel.

## Build the kernel snap

You are now ready to build the kernel snap.

1. Go to the directory with the cloned kernel repository.

* cd <kernel-source-repository>

1. Create an alias for the snapcraft.yaml file. This is only required if there are multiple YAML configuration files in the snap/local/ tree.

* ln -s snap/local/<project>.yaml snapcraft.yaml

1. (Optional) Add the sed command in the snapcraft.yaml file to set the Kconfig value CONFIG\_MODULE\_SIG\_ALL to n for your target architecture. This allows unverified modules to be loaded into the kernel and should only be set to n for local testing and development.

* For example, if the kernel snap is for an ARM64 device, set 'arm64': 'n':
* [...]  
  parts:  
   kernel:  
   override-build: |  
   [...]  
   # override configs  
   sed -i "s/^\(CONFIG\_MODULE\_SIG\_FORCE\).\*/\\1 policy\<{'arm64': 'n', 'armhf': 'n'}\>/" ${DEBIAN}/config/annotations  
   sed -i "s/^\(CONFIG\_MODULE\_SIG\_ALL.\*\)'arm64': 'y'\(.\*\)/\\1'arm64': 'n'\\2/" ${DEBIAN}/config/annotations  
   [...]

1. Build the kernel snap package.

* UC24 and UC22
* sudo snapcraft --build-for=arm64 --destructive-mode
* UC20
* sudo snapcraft --target-arch=arm64 --destructive-mode --enable-experimental-target-arch

1. You should get a <name>\_<version>\_<arch>.snap file in the kernel repository root, where:
   * <name> is the identified set in snapcraft.yaml
   * <version> is the kernel version
   * <arch> is the target architecture for the kernel snap
2. Copy the kernel snap to your target device and reboot into latest kernel to verify your changes.

* snap install --dangerous --devmode <name>\_<version>\_<arch>.snap
* ***Note****: Local snaps can only be installed if the Ubuntu Core image on the target device was created with a model assertion that specifies the “dangerous” grade.*