Run your Android apps on Linux

Swapping Places

Waydroid brings Android apps to the Linux desktop in a simple and effective way.

BY HARALD JELE

Emulators can be used to run applications from different operating systems in various constellations on Linux. The best-known candidates include Wine (Windows), DOSBox (DOS), and SNES (Nintendo games). But a counterpart for Android has been a long time coming, despite the clear proximity between the two systems. The current Android kernel is derived from a Linux kernel with long-term support (LTS). Despite many patches, there are basically more similarities between Android and Linux than differences. Having said this, running Android applications natively on Linux is complex and involves some tricky detailed work [1].

The makers of the free Waydroid [2] set themselves the task of integrating Android apps into the Linux universe as easily and flexibly as possible. When doing so, they relied on a proven approach and avoided reinventing the wheel. Anbox took a very similar path as early as in 2017, but the developers failed to follow up with a useful product. Anbox development was eventually discontinued in 2023.

Waydroid, like Anbox, is based on a container solution inside of which a session manager mounts and then launches an Android image. There are currently two images available, one with the central Google apps (`GAPPS`) and one without them (`VANILLA`). Both are descendants of LineageOS and are equivalent to an Android 11. They can be updated on the fly by an integrated update mechanism.

Installation

You can install the current Waydroid v1.4.1 on Ubuntu 22.04 LTS with just a few steps. The project website describe the details of the easy-to-follow procedure [3].

Listing 1: Waydroid Setup

```
01 $ sudo apt install curl ca-certificates -y
02 $ curl https://repo.waydro.id | sudo bash
03 $ sudo apt install waydroid wl-clipboard -y
04 $ sudo systemctl enable --now waydroid-container
05 $ sudo waydroid init -s SYSTEM_TYPE <Image>
```

In the first step, if not already present, you need to install two Ubuntu packages needed later (Listing 1, line 1). Then add the project's official repository to the local software sources (line 2); this will keep you up-to-date in the future. These sources are used to install the current version of the application (line 3) later.

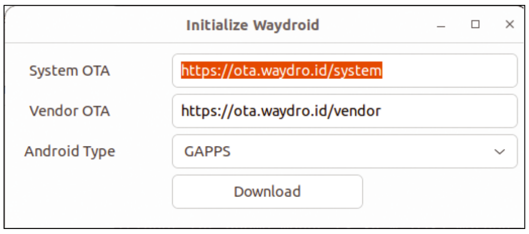
With this step, the required program components will now already exist in your Linux setup. Finally, you need to tell Ubuntu's system and session manager (systemd) to automatically start the Waydroid container at operating system boot time (line 4).

First Launch

When booting, Waydroid explores its configuration and determines prior to the initial launch that an Android instance has not yet been added. It then displays a graphical prompt, asking you to choose one of the two available instances (Figure 1).

You can choose either the Google-free `VANILLA` version or `GAPPS` for seamless integration with the Googleverse. If you change your mind later, type `init` at the command line to instruct Waydroid to load the other image and prepare it for mounting and booting (Listing 1, line 5).

However, Waydroid can only run one Android session inside a container so far. It makes sense to rename the previously loaded image before overwriting it by downloading the other one. The images for a Waydroid session reside in the `/var/lib/waydroid/images/` directory and are named `system.img` and `vendor.img`. You will want to rename these two files to keep them safe if you make any changes.


Initialize Waydroid

System OTA: https://ota.waydro.id/system
Vendor OTA: https://ota.waydro.id/vendor
Android Type: GAPPS

Figure 1: During the install, you need to select the Android image you want to use.

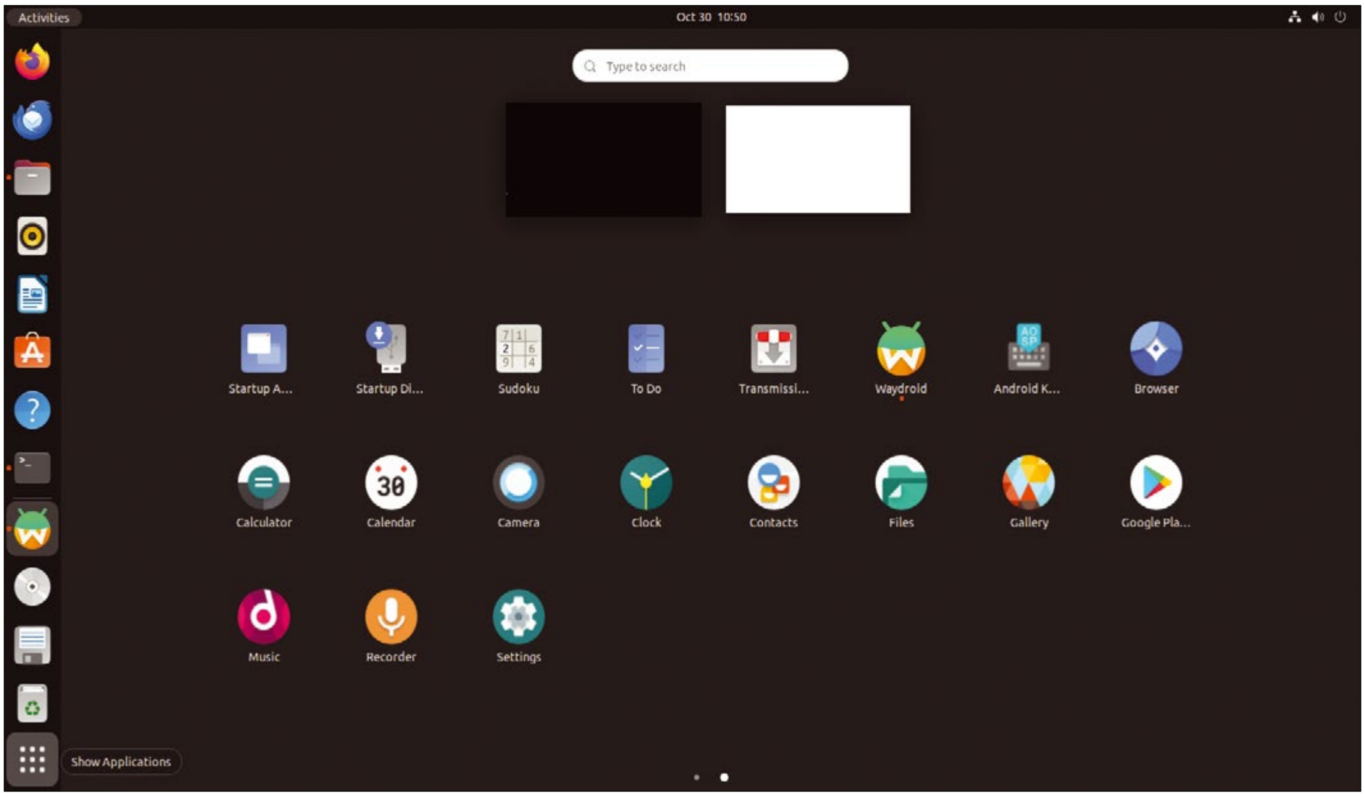


Figure 2: Android apps mingling with native Linux apps in the Ubuntu program launcher.

Using Applications

You are now ready to launch some initial Android apps. These apps blend in with the native Linux apps in the Ubuntu startup folder (Figure 2).

Once you have launched the `GAPPS` image, you need to register it with Google (Google Play certification) to fully enjoy Google Play. To do this, type `sudo waydroid shell` to start a Waydroid shell. In the shell, you then need to run the less than user-friendly command from line 1 of Listing 2 to discover the device ID and register it with Google [4] on *https://www.google.com/android/uncertified*.

Registration usually takes only a few seconds after you sign into your Google account. However, there are some posts on forums telling you that the procedure can take up to a few minutes. After completing the registration, you need to restart the Waydroid session (Listing 2, lines 3 and 4).

In the Android Universe

As mentioned before, the Ubuntu application launcher shows you the icons of any Android apps installed with the Google image alongside those of the native installation. For an overview of the apps that have been installed, you can run the `waydroid app list` command at the command line. You can also use the entries in this list to call an Android application from the command line. Lines 3 to 5 of Listing 3 show you an example of this that references the entry for a Google Docs app. You can launch the app directly in the terminal with the command from line 7.

Your options for launching Android apps include Google Play and the Google settings (*Settings | Apps*) like on a smartphone or tablet, calling the apps with Waydroid via the Ubuntu application launcher, or launching directly from a terminal. There are specific deployment scenarios for each of these options.

You can use Google Play to install additional apps if needed. F-Droid can also be integrated as an additional source in the usual way. On top of this, Waydroid provides an approach for setting up applications in APK file format directly (Listing 3, line 8).

Listing 2: Google Play Certification

```
01 $ ANDROID_RUNTIME_ROOT=/apex/com.android.runtime ANDROID_DATA=/data
     ANDROID_TZDATA_ROOT=/apex/com.android.tzdata ANDROID_I18N_ROOT=/
     apex/com.android.i18n sqlite3 /data/data/com.google.android.gsf/
     databases/gservices.db "select * from main where name = \"android_
     id\";"
02 [...]
03 $ waydroid session stop
04 $ waydroid session start
```

Listing 3: Launching Apps

```
01 $ waydroid app list
02 [...]
03 Name:          Docs
04 packageName:   com.google.android.apps.docs.editors.docs
05 categories:    android.intent.category.LAUNCHER
06 [...]
07 $ waydroid app launch com.google.android.apps.docs.editors.docs
08 $ waydroid app install <App>.apk
```

Problems

When installing new apps, you are likely to notice, sooner or later, that the Waydroid project still has a few rough edges. One of the most annoying problems is that rotating the display causes some applications to trip over their toes. Not all apps are suitable for operation in landscape mode. This is basically not a peculiarity of Waydroid, because apps like this will also fail if you run them natively on a cell phone or tablet.

What is annoying is the fact that you cannot reach half of the display with the mouse, in this case because Android only uses the width of the portrait format. In previous versions of Waydroid, this area simply remained black. In the current release, Waydroid does display the area, but it still cannot be used. Figure 3 shows a problematic application with the mouse pointer (which is very small in the figure) on the extreme right edge of the accessible area (within the red circle in Figure 3).

Basically, an application like this could be brought in line by rotating manually. However, manually changing the display geometry would then also affect all other apps. The simplest solution is to use the commands in Listing 4 to enable multi-window operation of the display, where all applications are displayed in portrait mode by default (Figure 4).

Currently, problems can still be caused by camera operations, the speakers, and the microphone. Waydroid is very keen on transparently passing the Linux standards through to

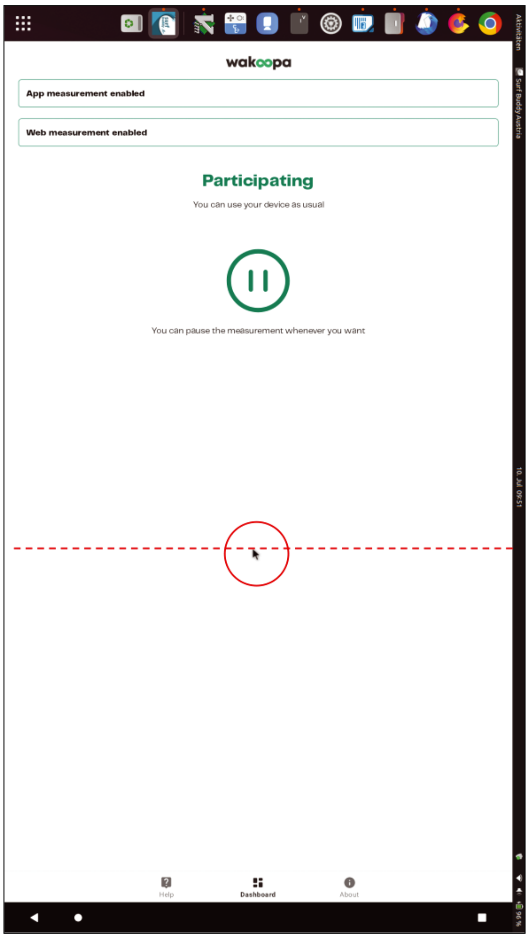
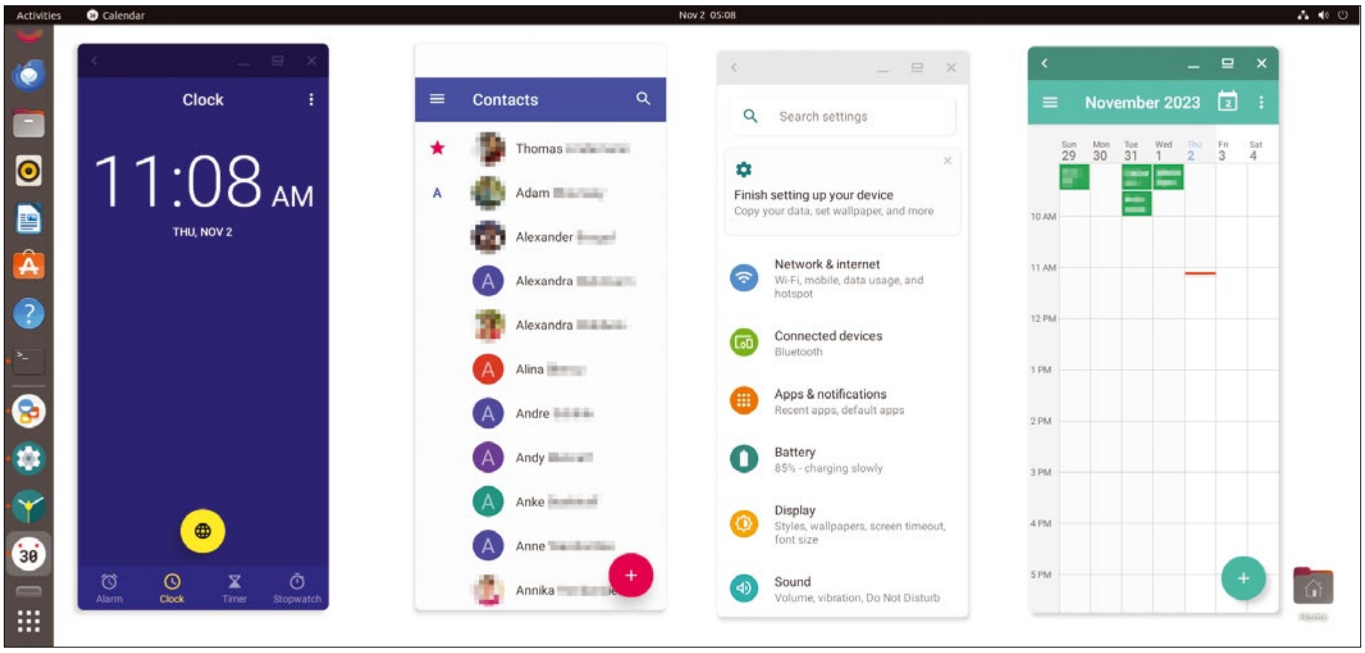


Figure 3: An example of an application that cannot be used in landscape mode.

Listing 4: Multi-Window Mode

```
$ waydroid prop set persist.waydroid.multi_
  windows true
$ systemctl restart waydroid-container.service
```

Figure 4: In multi-window mode, Waydroid displays all the Android apps in portrait mode.



the container. Nevertheless, it is still potluck as to whether the system responds properly to the hardware setup. The developers have been putting a great deal of work into the camera access for quite some time, so there should be some noticeable progress soon.

Command Line

Waydroid can be fully controlled and configured using the command line. But because the software works without any problems in many areas, many of these options remain more or less hidden. If you do want to take a closer look at the services Waydroid offers to the outside world, you will find detailed information about them in the Waydroid documentation [5]. For a first impression, you can try calling the Waydroid status report (Listing 5).

If there isn't an active session in the container, you can change this with `waydroid session start`, while `waydroid session stop` does what it says on the label. If you start an Android app with no active session, Waydroid automatically starts a session with the app.

If there isn't an active container, you can use

```
sudo waydroid container <option>
```

to change this. The options Waydroid accepts are `start`, `stop`, `restart`, `freeze`, and `unfreeze`. For the inquisitive or anyone wanting to troubleshoot an issue, it is useful to take a look at the `waydroid.log` file in `/var/lib/waydroid/`.

Listing 5: Waydroid Status

```
$ waydroid status
Session:         RUNNING
Container:       RUNNING
Vendor type:     MAINLINE
IP address:      192168240112
Session user:    admunix(1000)
Wayland display: wayland-0
```

Conclusions

The very flexible, open source Waydroid offers a useful approach to integrating Android applications into a Linux installation. Apps can be set up and used in the same way as on a smartphone. wl-clipboard [6] offers a neat way of exchanging data between the native Linux apps and the Android apps in the container.

Waydroid integrates well into the desktop of an Ubuntu installation; running Android apps is more or less the same as running native Linux apps. The project will very likely enable untroubled access to the camera, microphone, and speakers in the near future. This means that there is nothing stopping you from using your favorite apps from your smartphone or tablet on Linux. ■■■

Info

[1] Common Android kernel: *https://source.android.com/docs/core/architecture/kernel/android-common?hl=en*

[2] Waydroid: *https://waydro.id*

[3] Installing Waydroid: *https://waydro.id/#install*

[4] Google Play certification. *https://docs.waydro.id/faq/google-play-certification*

[5] Waydroid command line: *https://docs.waydro.id/usage/waydroid-command-line-options*

[6] wl-clipboard: *https://github.com/bugaevc/wl-clipboard*

The Author

Harald Jele is a member of staff at the University of Klagenfurt. He stumbled across Linux by happy coincidence in 1993 and has been using it on both servers and desktops ever since.