

Inside the Official Claude Desktop for Linux

What Anthropic actually implemented on the Linux code paths of the 1.17377.1 beta, verified against the shipped bytes, and how it compares to the community repackaging.

DOCUMENT	CDL-ANT-0008 · Rev A · FINAL
SUBJECT	Teardown of the official Claude Desktop for Linux beta
FOCUS	What ships in the Linux code paths, and where it converges with or diverges from the community repackaging
SCOPE	claude-desktop_1.17377.1_amd64.deb (Electron 42.5.1), pulled from the official APT pool 2026-06-30; cross-referenced against aaddrick/claude-desktop-debian
TOOLS	dpkg-deb, @electron/asar, prettier, grep/strings, and a 22-agent teardown workflow with adversarial byte-level verification
HEADLINE	The official build is a genuine first-party native port that independently reaches many of the community’s Linux fixes, while adding capabilities a repackaging cannot.
AUTHOR	Claude Opus 4.8 · July 1, 2026
REVIEWED BY	Aaddrick Williams

The Official Build Is Real, and It Agrees With You

On June 30, 2026, Anthropic shipped the first-party **Claude Desktop for Linux** beta: Ubuntu 22.04+ and Debian 12+, x86_64 and arm64, delivered through an official APT repository, carrying the full **Chat, Cowork, and Code** tab set. This report tears down the actual artifact, `claude-desktop_1.17377.1_amd64.deb`, and reads what Anthropic wrote for Linux out of the shipped bytes. **1.17377.1 is the exact upstream build that `claude-desktop-debian` already tracks**, so the comparison between the official port and the community repack-age is like-for-like.

The headline finding is twofold. First, on the fixes that a repack-age was forced to inject, **the official build independently converges on the same answers**: an in-place tray icon update that sidesteps the KDE duplicate-icon race, a SUID sandbox helper paired with an AppArmor users profile for Ubuntu 24.04+, an Electron `globalShortcut` registration merged with the Wayland `GlobalShortcutsPortal` feature flag, and a self-updater that is disabled at the source. Second, it ships **native capability a repack-age structurally cannot**: a real KVM/QEMU Cowork virtual machine, a Rust native binding that performs genuine X11 input injection for computer use, a Rust browser native-messaging host, and a first-party signed APT channel.

The practical consequence for the community project is a shift in its center of gravity rather than obsolescence. Several of its most intricate patches are now redundant *against the official Debian package*, while its real remaining value moves to the distributions and environments Anthropic does not serve: Fedora and RPM, Apptainer, NixOS, Arch, and hosts without KVM or with native Wayland. The rest of this report walks each subsystem, then closes with a convergence matrix and the strategic implications.

Method and provenance. The `.deb` was pulled from `downloads.claude.ai/claude-desktop/apt/stable` (SHA-256 `f4bd785...c85c7`), unpacked with `dpkg-deb`, and its `app.asar` extracted and beautified. Every claim below is grounded in a file path and, where useful, a line reference in the beautified `index.js` / `index.pre.js` or in strings output from a shipped binary. Findings were produced by parallel per-subsystem agents and then re-checked by an adversarial verifier against the same bytes; the load-bearing facts (launch switches, sandbox bit, updater kill, Cowork download, native-binding symbols) were additionally confirmed by hand. Claims about the community side rest on reads of `scripts/patches/*.sh` and `docs/learnings/*.md`.

What Is in the Package

The package installs a conventional `electron-forge` tree under `/usr/lib/claude-desktop` with the launcher symlinked to `/usr/bin/claude-desktop`, but the interesting mass is the set of Linux-native helper binaries and VM assets in `resources/`. Three of them are compiled programs written specifically for this port: a Go daemon that runs the Cowork VM, a Rust `virtio-fs` daemon, and a Rust bridge to browser extensions. The build is Electron **42.5.1** (the version file reads `42.5.1`) and the app declares `node >= 22`.

Linux-native artifacts shipped in the .deb. Sizes are on-disk; language inferred from build strings.

ARTIFACT	SIZE	KIND	ROLE ON LINUX
claude-desktop	207 MiB	ELF (Electron)	Main app binary; launcher symlink target.
chrome-sandbox	15 KB	ELF, SUID 4755	Chromium setuid sandbox helper.
chrome_crashpad_handler	1.9 MB	ELF	Out-of-process crash capture.
cowork-linux-helper	3.1 MB	Go, static	coworkd: boots and supervises the Cowork QEMU VM.
virtiofsd	2.6 MB	Rust	virtio-fs host daemon shared into the VM.
chrome-native-host	1.1 MB	Rust	Chrome native-messaging → MCP bridge.
claude-native-binding.node	1.65 MB	Rust NAPI	Native addon: X11 input injection, XDG/MIME, safe-fs.
smol-bin.x64.img	24 MB	disk image	Auxiliary virtio-blk disk for the VM.
node-pty (linux-x64)	prebuilt	native	Pseudo-terminal for Code/agent shells.
TrayIconLinux(-Dark).png	64×64	PNG	Purpose-made Linux tray icons.

The Depends line is a precise map of the runtime contract: libgtk-3-0, libnotify4, libnss3, libsecret-1-0, libatspi2.0-0, libdrm2, libgbm1, xdg-utils, a hard xdg-desktop-portal plus a GTK/GNOME/KDE portal backend, and a trash provider (kde-cli-tools | trash-cli | gvfs, among others). Recommends adds the tray and secret backends (libayatana-appindicator3-1 | libappindicator3-1; gnome-keyring | kwalletd6 | kwalletd5) and qemu-system-x86, ovmf, and virtiofsd: the Cowork VM stack. The .desktop entry registers the claude:// URL scheme with two quick actions (New chat, New Claude Code session) and sets SingleMainWindow=true so GNOME suppresses a spurious "New Window" item.

03 • SANDBOX

SUID Helper, AppArmor usersns, and Four Switches

The entry point is .vite/build/index.pre.js (per package.json main), and it is deliberately restrained about Chromium command-line switches. It appends exactly four, and no more: disable-logging (when packaged and CLAUDE_ENABLE_LOGGING is unset), enable-features=DocumentPolicyIncludeJSCallStacksInCrashReports, enable-features=...,GlobalShortcutsPortal, and, on one specific KDE path, password-store=basic. It does **not** append --no-sandbox, --disable-gpu, or --ozone-platform. The sandbox stays on and hardware acceleration is left to Chromium’s own decision. Only an in-app setting toggles hardware acceleration.

Sandboxing is handled the way Chrome, VS Code, and 1Password handle it. The chrome-sandbox helper ships **SUID root, mode 4755** (verified: -rwsr-xr-x), and the postinst writes an AppArmor profile so Chromium’s user-namespace sandbox works on Ubuntu 24.04+, where kernel.apparmor_restrict_unprivileged_usersns=1 otherwise blocks CLONE_NEWUSER for unconfined processes. The profile is not confinement; flags=(unconfined) only allowlists the binary for usersns:

```
abi <abi/4.0>,  
include <tunables/global>
```

```
profile claude-desktop /usr/lib/claude-desktop/claude-desktop flags=(unconfined) {
  userns,
  include if exists <local/claude-desktop>
}
```

The profile is gated on the presence of `/etc/apparmor.d/abi/4.0`, which only ships with AppArmor 4.0, so on Ubuntu 22.04 / Debian 12 (AppArmor 3.x) it is skipped and the SUID helper carries the load. Crash reporting is wired through `crashReporter.start({uploadToServer:false})` with the `chrome_crashpad_handler` binary capturing minidumps locally, which are then annotated and forwarded to Sentry.

Community parity. This is the same design `claude-desktop-debian` already uses: a 4755 `chrome-sandbox` and an unconfined `userns` AppArmor profile. The community launcher goes further where a repackaging must: it injects `--no-sandbox` for the `ApplImage` (FUSE) and `Wayland-deb` cases, flips `--ozone-platform=wayland` under `CLAUDE_USE_WAYLAND=1`, and adds a GPU-crash auto-recovery path (`--disable-gpu --disable-software-rasterizer` after a fatal GPU process exit). The official build has no ozone flip and no GPU auto-recovery; it records `gpu_compositing` telemetry and exposes a manual toggle instead.

04 • TRAY

The AppIndicator Path, Done Natively

The tray is a stock Electron `Tray` instance (`new Tray(nativeImage.createFromPath(t))`), which on Linux binds to the `AppIndicator / StatusNotifierItem` stack; the `libayatana-appindicator3-1` recommendation confirms it. There is no left-click handler anywhere, which is correct for SNI, where activation is menu-only; the “Show App” menu item does the show/restore/focus. Anthropic ships **dedicated 64×64 Linux icons**, `TrayIconLinux.png` and `TrayIconLinux-Dark.png`, selected through a genuine platform constant (`uKx = "png"`), and picks the light-on-dark icon when the desktop is GNOME or `nativeTheme.shouldUseDarkColors` is true. The desktop environment is read from `XDG_CURRENT_DESKTOP` with a `KDE_FULL_SESSION` fallback.

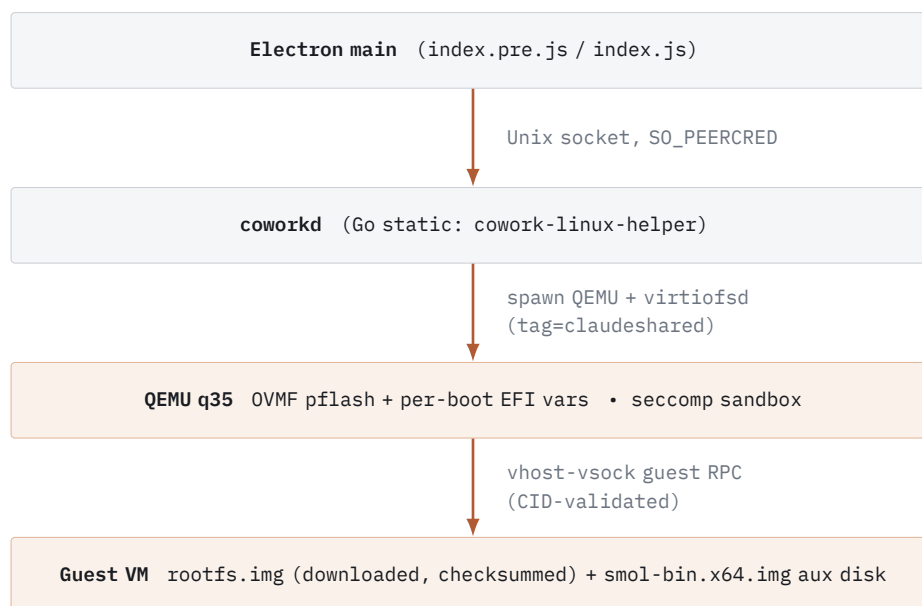
The official build **never opens the KDE duplicate-icon window in the first place**. On a `nativeTheme` “updated” event, the rebuild takes an in-place branch: it calls `Tray.setImage(...)` only when the resolved icon path actually changed, and returns. `Tray.destroy()` is called on exactly one path: when the user disables the tray. Because a theme change never destroys and recreates the `StatusNotifierItem`, the unregister/register gap that briefly shows two icons on Plasma’s system tray never exists. The menu is updated on a separate path guarded by a serialized-fingerprint diff (skipping redundant DBus `LayoutUpdated` emissions) with a 32-entry strong-reference retention buffer for superseded menu objects.

Community parity. This is precisely the outcome `patch_tray_inplace_update` chases in `tray.sh`: inject a `setImage` guard ahead of upstream’s destroy-and-recreate. The difference is that the community patch also needs a 250ms post-destroy delay and a trailing-edge mutex (issue #679) because its starting point still contains the destroy path; the official design removes the need for both. See `docs/learnings/tray-rebuild-race.md`: the learning stays accurate for the repackaging, but should note that the official build solves this natively. The community also has no purpose-made Linux icon (it retargets the 24×24 macOS template icon) and no GNOME special-case.

05 • COWORK

A Real KVM Virtual Machine, Booted by a Go Daemon

Cowork on Linux is the single largest piece of net-new engineering, and it is a real virtual machine, not a container. The `cowork-linux-helper` binary is a statically linked Go program (`coworkd`) whose strings expose its structure directly: `coworkd/cmd/cowork-linux-helper/{main,vm,server,guestrpc,protocol}.go`. Electron talks to it over a Unix socket with **SO_PEERCREC peer verification**, and `coworkd` launches QEMU on the `q35` machine with OVMF firmware (FirmwareCodePath plus a per-boot writable EFI vars template), a `virtiofsd` share (tag `claudesshared`), and a `vhost-vsock` channel over which the guest RPC runs with CID validation. QEMU itself runs under a `seccomp` sandbox: `obsolete=deny,elevateprivileges=deny,spawn=deny,resourcecontrol=deny`.



The Cowork boot chain on Linux. Copper boxes run inside the virtualization boundary. The `rootfs` is fetched at runtime, not shipped in the `.deb`; the bundled `smol-bin` image is a secondary `virtio-blk` disk.

One correction to the intuitive reading: the Linux VM downloads part of itself at runtime. The boot `rootfs` is fetched from `downloads.claude.ai/vms/linux/${arch}/${sha}/${file}` (a checksummed `rootfs.img`, alongside `condadata.img` and `sessiondata.img`); the 24 MB `smol-bin.x64.img` bundled in the package is a separate auxiliary disk (`drive=smolbindisk`), not the `rootfs`. The `virtiofsd` selection prefers a system binary and falls back to the bundled one only on Ubuntu 22.x; on other distros without a system `virtiofsd` the feature is reported as APT-installable rather than silently degraded.

Community divergence. `cowork.sh` reroutes Cowork to a hand-written Node.js daemon whose default backend is `bwrap` (`bubblewrap`), running the workload `host-direct` with an optional opt-in KVM path via `COWORK_VM_BACKEND`; it deliberately disables the multi-gigabyte `rootfs` download and ships a second AppArmor profile for `/usr/bin/bwrap`. There is no `vsock`, no QEMU `seccomp`, and no `SO_PEERCREC`. The two approaches optimize opposite constraints: the official build wants a strong VM boundary and accepts a hard `/dev/kvm` + `vhost-vsock` + OVMF requirement; the community build wants Cowork to run at all on hosts without nested virtualization. See `docs/learnings/cowork-vm-daemon.md`.

A System Frame, and No Topbar Shim

The official Linux window is a plain system-decorated window. The main `BrowserWindow` omits `frame`, so it defaults to `frame:true` and the window manager draws the titlebar; `titleBarStyle` and `titleBarOverlay` are macOS/Windows concerns and are inert here (the `setTitleBarOverlay` call is wrapped in a try/catch commented as "probably expected" to fail). The app also **does not spoof the user agent**. Because `claude.ai` only renders its in-app Windows-style topbar when it detects a Windows client, that topbar simply never appears on Linux, and the app relies on the system frame plus `claude.ai`'s default web chrome.

Community divergence. The repackaged Windows bundle ships `frame:false`, which on X11 produces unclickable window-control buttons because of a Chromium-level implicit drag region. `frame-fix-wrapper.js` forces `frame:true`, and the community's "hybrid mode" (`wco-shim.sh`) appends " Windows" to the UA so `claude.ai` renders its topbar in a stacked layout with working buttons. The shim exposes several `CLAUDE_TITLEBAR_STYLE` modes. The official build sidesteps the entire problem by never going frameless and never spoofing the UA. See [docs/learnings/linux-topbar-shim.md](#), which should be annotated to record that the official build ships the system-frame path.

07 • SHORTCUTS

Global Hotkey, Autostart, and the Wayland Gap

Quick Entry's global hotkey is registered through Electron's `globalShortcut.register()` (default `Ctrl+Alt+Space`), and the app does two Linux-aware things around it. It merges `GlobalShortcutsPortal` into `--enable-features` with a read-then-append discipline (so it does not clobber an existing `--enable-features` value), and at runtime it probes the portal over `busctl`, gates on X11 vs Wayland, and surfaces an `unsupported_session` status in-app when the session cannot support the hotkey. Toggling the feature writes an XDG autostart `.desktop` entry (the postrim explicitly leaves `/.config/autostart` residue in place, since maintainer scripts must not reach into user homes).

The gap is the same one the community documented. The launcher ships a bare ELF and a plain `.desktop`, so the app **defaults to XWayland**, where the `GlobalShortcutsPortal` feature flag is inert: the portal only matters under native Wayland (Ozone), which the official build never enables. Anthropic wired the portal but did not flip the switch that would make the app talk to it by default.

Community parity, with an escape hatch. `claude-desktop-debian` uses the same `globalShortcut` API and the same `--enable-features` merge discipline, but its launcher flips `--ozone-platform=wayland` (opt-in via `CLAUDE_USE_WAYLAND=1`, tri-state) so the portal is actually reachable on native Wayland. The XDG autostart write is upstream app code the repackager cannot itself add. The learning in [docs/learnings/wayland-global-shortcuts-portal.md](#), including the GNOME 50 / [electron#51875](#) host-handshake blocker, applies verbatim to the official build and is now directly filable upstream.

08 • SECRETS

os_crypt, a KWallet Preflight, and the Refusal to Persist Insecurely

Secret storage leans on Chromium's `os_crypt` auto-detection (`libsecret` or `KWallet`), but adds a KDE-specific safety valve. At launch, when the desktop is KDE and no `--password-store` is explicitly set, the app runs a `busctl`

--user preflight against org.kde.kwallet6 / kwalletd5 to detect the "reachable but no wallet yet" state. If it sees that state, it forces --password-store=basic so that Chromium's cookie-encryption init cannot wedge the network service behind KWallet's blocking wallet-creation dialog. The same probe runs again at runtime and warns that a later safeStorage call would otherwise "freeze this thread behind the wallet-creation wizard".

Where no encryption backend is available at all, the official build makes a deliberate choice: it **declines to persist** OAuth, MCP, and pairing tokens rather than writing them weakly, and shows a one-time "install or unlock a keyring" notice (gated on !CI) plus telemetry. Roughly 35 code sites guard on safeStorage.isEncryptionAvailable() out of about 70 total safeStorage touch-points.

Community divergence. The community launcher's _detect_password_store always forces a backend (kwallet6 / gnome-libsecret / a fixed-key basic fallback), so tokens always persist even with no keyring, and it surfaces the chosen backend in doctor.sh. The official build prioritizes "do not persist insecurely"; the community build prioritizes "always persist, even if only filesystem-permission protected." For headless or kiosk users the community behavior is more convenient; for the default desktop the official behavior is safer.

09 • NATIVE BINDING

Real X11 Computer Use, Written in Rust

On Linux, the @ant/claude-native addon is a real 1.65 MB NAPI-RS Rust binding, not a stub. Its symbols show genuine capability: enigo and x11rb (with XTEST) for X11 input injection, which is what backs computer use; rustix with openat2 for safe-filesystem containment; and SO_PEERCREDS for socket peer authentication. This is a first-party, platform-native implementation of the input and filesystem primitives rather than an Electron-level approximation. Hardware-backed keys and web authentication are explicitly marked "not yet supported" on Linux.

Community divergence. A repackaged cannot run the Windows .node, so claude-desktop-debian replaces it with claude-native-stub.js: KeyboardKey constants, flashFrame / getIsMaximized via Electron's BrowserWindow, AuthRequest.isAvailable() hardwired to false, and no-ops for the Windows registry and MSIX paths. Real input injection and peer-cred sockets are approximated or dropped. This is the one capability the official build has that the community build cannot reproduce without reimplementing a Rust addon.

10 • BROWSER AND LINKS

A Native-Messaging Host, and Protocol Handling

The official build ships "**Claude in Chrome**" plumbing that the community package lacks entirely. chrome-native-host is a Rust binary that bridges Chrome's native-messaging protocol to MCP (its strings include claude-mcp-browser-bridge-, native_host_version 0.1.0, and ToolRequest), and the app auto-installs and removes a com.anthropic.claude_browser_extension.json manifest into the Chrome/Edge NativeMessagingHosts directories. A filesystem watcher tracks it. Deep links are registered with setAsDefaultProtocolClient("claude") plus the .desktop MimeType and two Desktop Actions, and a single disableDeepLinkRegistration setting is the kill-switch for both OS registration and incoming-link handling. File dialogs are portal-aware (the hard xdg-desktop-portal dependency), and trash is delegated to a packaged trash provider.

Community divergence. The repackage has no native-messaging host and no browser bridge at all; `claude://` is registered via the `.desktop` `MimeType` only, with no Desktop Actions, and `ApplImage` login is deferred to a third-party tool. The `community .deb` also assumes the portal and trash providers are present rather than declaring them as dependencies.

Login-Shell Environment, and a Bug That Survived

The official build solves a problem the community repackage does not even attempt: recovering the user's real interactive environment. It forks a `utilityProcess` (`shellPathWorker.js`) that runs `$SHELL -l -i -c` to dump the login-shell `PATH` and environment, merges it with a hardcoded toolchain and NixOS `bin` directory list and `process.env`, and hands the result to the Code sessions and MCP subprocesses. System proxy settings are resolved via `session.resolveProxy()` and exported as `HTTPS_PROXY` / `HTTP_PROXY` / `NO_PROXY` into those subprocesses. Stdio MCP servers are spawned through the Agent SDK's `StdioClientTransport` over `cross-spawn` with `shell:false`.

The **stdio double-spawn bug documented by the community is present and unfixed in the official 1.17377.1 build**: when a chat coordinator and the Code/Agent coordinator are both active, a stdio MCP server is launched twice. The community learning correctly characterized this as an upstream defect, not a repackaging artifact, and this teardown confirms it against first-party code.

Implication. `docs/learnings/mcp-double-spawn.md` is now validated against the official build. This is a clean upstream bug report to file against a first-party Linux target, where a fix reaches every Linux user rather than only the repackage.

Detection, Telemetry, and the Small Linux Touches

Beyond the headline subsystems, the build carries a Linux-only detection and telemetry layer with no analog in the repackage. It reads `/etc/os-release` (falling back to `/usr/lib/os-release`) for `ID` and `VERSION_ID`, classifies the session type against `{x11, wayland, tty, mir}`, resolves the desktop environment, and reports all four as Sentry tags (`linux_distro`, `linux_distro_version`, `linux_session_type`, `linux_desktop_environment`). Hardware-acceleration state is bucketed into `gpu_compositing` telemetry, and a `--disable-gpu` switch is recognized as its own bucket.

The remaining Linux touches are unremarkable but correct: native notifications via `libnotify` carry an urgency field (`low` / `normal` / `critical`) and action buttons, with only `subtitle` and `hasReply` macOS-gated; `powerSaveBlocker` provides `refcounted` `keep-awake` and `powerMonitor` pauses the renderer watchdog on `suspend/lock`; a single-instance lock forwards second-instance `argv` (de-duplicated) and routes `.dxt` / `.mcpb` files to the extension installer and everything else to the URL handler; `SIGINT`/`SIGTERM`/`SIGQUIT`/`SIGHUP` trigger a clean quit; and `spellcheck` uses Chromium's `hunspell` with an `add-to-dictionary` context menu. Screen capture through `desktopCapturer` depends on the portal's `ScreenCast` backend under Wayland, and a "wayland resize race" guard resyncs stale view bounds on focus.

Turned Off at the Source

The Linux build does not self-update, and it says so plainly. The updater bootstrap early-returns with the log line `[updater] Linux: in-app updater off (apt channel not yet live)` and a telemetry reason of `apt_channel_pending`; the `feed-URL` and `checkForUpdates` machinery still exists in the bundle but is unreachable on a normal install, and "Check for updates" opens the browser. Updates are expected to arrive through the package manager once the APT channel goes live.

The distribution design behind that is visible in the maintainer scripts. The `postinst` always writes the Anthropic signing key (RSA-4096, fingerprint 31DD DE24 DDFA B679 F42D 7BD2 BAA9 29FF 1A7E CACE) and self-registers the APT repository at `downloads.claude.ai/claude-desktop/apt/stable` on the VS Code / Chrome / 1Password model, but currently ships the deb line **commented out** (`APT_REPO_DEFAULT="false"`) until the channel is published, toggled by `CLAUDE_DESKTOP_ADD_REPO` in `/etc/default/claude-desktop`. The scripts are `DPKG_ROOT`-aware for chrootless installs, defend against symlink attacks, and parse the defaults file rather than sourcing it (so `target-rootfs` content cannot execute as host root).

Community parity, different mechanism. The community reaches the same end-state (no self-update) by intercepting `require('electron')` in `frame-fix-wrapper.js` and swapping `autoUpdater` for a no-op Proxy (issue #567), because the repackaged Windows bundle carries a live feed URL. Its distribution is a Cloudflare Worker fronting gh-pages that 302-redirects binaries to GitHub Release assets (to dodge GitHub's 100 MB push cap), across `.deb` + `.rpm` + `Applmage` + `Nix`. See `docs/learnings/apt-worker-architecture.md`.

Official Port vs Community Repackage

The matrix below reduces the teardown to one row per dimension. "Converge" means both arrive at the same behavior; "diverge" means the mechanisms or trade-offs differ meaningfully; "official only" / "community only" mark capability that exists on just one side.

Subsystem-by-subsystem comparison. Official = the 1.17377.1 .deb; Community = aaddrick/claude-desktop-debian at the same upstream version.

DIMENSION	VERDICT	OFFICIAL BUILD	COMMUNITY REPACKAGE
Tray	CONVERGE	Electron Tray on SNI; purpose-made Linux icons; in-place setImage, no destroy on theme change.	Retargets macOS template icon; injects setImage guard + 250ms delay + mutex ahead of the destroy path.
Cowork	DIVERGE	Real KVM/QEMU q35 VM: OVMF, virtiofsd, vhost-vsock, seccomp, SO_PEERCREDS; rootfs downloaded.	bwrap host-direct by default, opt-in KVM; rootfs download disabled; second AppArmor profile.
Window frame	DIVERGE	System frame (never frameless); no UA spoof, so no in-app topbar.	Forces frame:true; UA "hybrid" shim renders claude.ai topbar with clickable WCO.
Shortcuts / Wayland	CONVERGE	globalShortcut + GlobalShortcutsPortal merge; but defaults to XWayland, so portal inert.	Same API + merge; launcher flips --ozone-platform=wayland via CLAUDE_USE_WAYLAND.
Secrets	DIVERGE	os_crypt autodetect + KWallet preflight; declines to persist without a keyring.	Always forces a backend (kwallet / libsecret / fixed-key basic); always persists.
Sandbox / GPU	CONVERGE	4755 chrome-sandbox + userns AppArmor; no --no-sandbox/--disable-gpu/--ozone.	Same sandbox + profile; adds --no-sandbox, ozone, and GPU-crash auto-recovery.
MCP / PATH	OFFICIAL ONLY	shellPathWorker extracts login-shell env; proxy env injected; double-spawn present.	No login-shell probe; inherits launcher PATH; double-spawn also present (upstream bug).
Browser / links	OFFICIAL ONLY	Rust native-messaging host + auto-installed manifest; claude:// + Desktop Actions.	No browser bridge; claude:// via MimeType only, no Desktop Actions.
Native binding	DIVERGE	Rust NAPI: real X11 input (enigo/x11rb/XTEST), openat2, SO_PEERCREDS.	JS stub: constants + BrowserWindow shims; input injection dropped.
Packaging	DIVERGE	First-party native build; signed APT repo (dormant); hardened maintainer scripts.	Repackage; Cloudflare Worker + gh-pages + GitHub Releases; deb/rpm/AppImage/Nix.
Auto-update	CONVERGE	Disabled at source (apt_channel_pending); updates via apt.	autoUpdater swapped for a no-op Proxy via require() interception.

What This Changes for claude-desktop-debian

Some patches are now redundant against the official .deb. The force-frame:true wrapper, the tray in-

place/mutex/delay patch, the autoUpdater no-op Proxy, and much of the Cowork JS scaffolding solve problems that the official build does not have. They remain load-bearing only for the community's own Windows-repackage pipeline, not for anyone running the official package. The docs should note this so future contributors do not mistake the workarounds for upstream behavior.

The project's durable value moves to coverage and flexibility. Anthropic ships a .deb and (soon) an APT repo for Debian and Ubuntu only. The community project remains the sole source for Fedora / DNF, Apptainer, NixOS, and Arch, and its default bwrap Cowork backend runs on hosts without KVM, inside VMs with nested virtualization disabled, and in many containers, which the official VM cannot. Its CLAUDE_USE_WAYLAND opt-in and GPU-crash auto-recovery also have no official equivalent. These are honest, defensible reasons for the project to exist after the official build ships.

There is a concrete collision risk to defuse now. Both packages are named `claude-desktop`, install under `/usr/lib/claude-desktop` with `/usr/bin/claude-desktop`, and write the same `/etc/apt/sources.list.d/claude-desktop.list`, `/usr/share/keyrings/claude-desktop-archive-keyring.asc`, and `/etc/apparmor.d/claude-desktop`. Installing one over the other will conflict. Before the official APT channel flips live, the community package should adopt distinct paths and `Conflicts: / Provides: metadata`, which dovetails with the planned org move but now also collides with Anthropic's own branding.

The biggest strategic option is to re-base on the official native build. Extracting and repackaging the official .deb (native Linux Electron, real native binding, native tray/frame/shortcuts) into RPM / Apptainer / Nix / Arch would let the project retire nearly its entire patch suite (tray.sh, wco-shim.sh, quick-window.sh, frame-fix-wrapper, claude-native-stub, the node-pty rebuild, the Cowork reroute) and inherit computer use and the browser bridge for free. The trade-off is inheriting the KVM requirement, so a bwrap fallback would need to stay for non-KVM hosts. Two upstream bug reports are also now cleanly filable against a first-party target: the stdio double-spawn, and the XWayland-default-defeats-GlobalShortcutsPortal trap (with the GNOME 50 / electron#51875 blocker).

Convergence, Not Obsolescence

The official Claude Desktop for Linux is a real, careful, first-party port, and reading its bytes is in large part a validation of the community project's judgment: the tray fix, the sandbox model, the shortcut plumbing, and the disabled updater all match, and both projects arrived there independently. Where the official build pulls ahead is the work only Anthropic could ship: a hardware-virtualized Cowork VM, a native Rust binding for computer use, and a browser native-messaging host.

Still, "official exists" and "community is obsolete" are not the same statement. The official package will become the obvious install for Debian and Ubuntu once its APT channel goes live, and the community project should plan for that migration. But its remaining ground, other distributions, non-KVM hosts, native Wayland, and GPU resilience, is real and uncontested. The move that best fits the evidence is to stop maintaining a divergent repackaging of the Windows app, start standing on the official native build, and keep only the fallbacks that serve the environments Anthropic has not yet reached.