Episode Book

EPISODE #5969

Ubuntu 26.04 LTS: Is Staying on LTS Still a Trade-Off?

PUBLISHED	14 October 2026 at 15:13 UTC
DURATION	30:19
LISTEN	myweirdprompts.com/episode/ubuntu-2604-lts-trade-offs-04441/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.4
SOURCES	12 cited by the research (listed at the end)
BOOK GENERATED	19 October 2026 at 17:42 UTC

SYNOPSIS

Ubuntu 26.04 LTS landed in April 2026, and the first point release arrived in late September. So is staying on the LTS track still a meaningful technical compromise, or have point releases, hardware enablement stacks, and stable release updates closed the gap? This episode walks through how Canonical's maintenance model actually works — five years of standard support, up to fifteen with Ubuntu Pro, and the escape hatches that let fast-moving software bypass the freeze. Then it examines where staleness still shows up: the Wayland-only desktop, the application isolation layer, and the one place the model genuinely gets tested — AMD GPU compute, where ROCm is now native in the Ubuntu archive and managed by apt. If you've ever wondered whether to upgrade or hold steady, this is the breakdown.

TRANSCRIPT



Corn

The temptation is always the same. You're on a stable release, everything works, and then you read about a newer kernel, a newer driver, a newer version of something you use every day. And you start thinking about how much better your life would be if you just upgraded.



Herman

Right, and then you do it, and you spend the weekend fixing things that weren't broken.



Corn

Daniel's been running Linux desktops for almost twenty years, mostly Ubuntu, and he just moved to 26.04 LTS. He's planning to stay on the LTS track this time. But he wants us to be honest about what that costs him. His question is whether staying on LTS is still a meaningful technical compromise this year, or whether the mechanisms around it, the point releases and the hardware enablement stacks and the container stuff, have basically closed the gap.



Herman

It's a good time to ask it, because 26.04 shipped in April, and the first point release landed at the end of September. So the picture is complete now. The launch stack is settled, the hardware enablement kernel is available, and we can actually see what Canonical is doing with the maintenance model.



Corn

So the question splits into four. How the maintenance model works. How much freshness you actually give up. Whether AMD GPU inference is a special case that breaks the whole argument. And when it makes sense to leave LTS behind.



Herman

Let's start with the model, because everything else depends on understanding what LTS actually promises. Ubuntu ships every six months, April and October. Every even-year April release is an LTS. So 24.04, 26.04, 28.04, and so on. The interim releases between them get nine months of support. That's it. If you install 26.10 in October, you're patching it until July and then you have to move again.



Corn

Nine months.



Herman

The LTS gets five years of standard security maintenance on the Main repository. For 26.04, that runs to May 2031. And then Ubuntu Pro extends that, Expanded Security Maintenance, out to May 2036, and there's a Legacy add-on that takes it to May 2041. Fifteen years total.



Corn

If you'd installed 26.04 the day it came out and never touched it again, you'd be getting security patches until 2041.



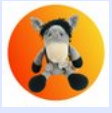
Herman

On the Main repository, through Pro. And Pro is free for personal use on up to five machines. That's the part people miss. They hear "Ubuntu Pro" and assume it's an enterprise subscription, but for a home desktop it costs nothing. You get the Universe coverage, the extra ten years of patching, and Livepatch on top.



Corn

What does Livepatch actually buy you on a desktop?



Herman

Kernel patches without rebooting. For a server that's obvious. For a desktop it's less dramatic, but if you're the kind of person who leaves the machine running for weeks at a time, it means kernel security fixes just land in the background and you never see the reboot prompt. It's one of those things that sounds like marketing until you've had a laptop sitting on a kernel CVE for three weeks because you kept deferring the restart.



Corn

Right. So the LTS support window is not the constraint people think it is. Five years is long. Fifteen with Pro is longer than most hardware lasts.



Herman

And the point releases matter for a different reason. 26.04.1 arrived on the twenty-ninth of September. A point release isn't a new version of Ubuntu. It's accumulated stable updates rolled into refreshed ISO files, plus refreshed hardware support. So if you install from the 26.04.1 image today, you're not applying a year of patches after install. You're getting a snapshot that already includes all of them.



Corn

And it refreshes the hardware support.



Herman

That's the part that actually changes what your machine can do. The point release picks up the newer kernel and the newer graphics stack that have been backported through the hardware enablement programme. Which is the mechanism that most people don't know about and it's the one that answers Daniel's question directly.



Corn

Explain the HWE stack, because this is the thing that quietly makes LTS work on new hardware.



Herman

Hardware Enablement. The idea is simple. LTS ships with a kernel. That kernel is supported for the life of the release. But new hardware comes out, and the kernel you shipped eighteen months ago doesn't have the drivers for it. So Canonical backports a newer kernel and a newer X or Wayland graphics stack from a later interim release, into the LTS, as an optional stack. You get it through the normal update mechanism. It sits alongside the original kernel, which keeps getting security patches, and you choose which one you're tracking.



Corn

And on desktop installs, you're tracking HWE by default.



Herman

Desktops default to HWE. Server installs default to the general availability kernel, and you opt into HWE if you want it. That default tells you something. On a desktop, Canonical assumes you want the newer kernel and you'll take the risk that comes with it. On a server, they assume the opposite.



Corn

The reverse of what you'd expect if you thought LTS meant frozen.



Herman

Completely. And the kernel lifecycle table shows it. Linux 7.0 ships in 26.04.0 as the general availability kernel, and it's supported to May 2031. The same 7.0 kernel shipped as the HWE kernel for 24.04 in the point five release in July, supported to May 2029. So two different LTS releases are running the same kernel, with different support windows, because they arrived at it from different directions.



Corn

So a 24.04 user on the HWE stack is on Linux 7.0. That's the kernel from this year's LTS, running on the previous LTS.



Herman

Which is the whole argument. If you're on 24.04 and you bought a laptop with hardware that only works on a recent kernel, you're already there. You didn't wait for 26.04. The HWE stack brought it to you.



Corn

You can check what you're running, presumably.



Herman

There's a tool, hwe-support-status with the verbose flag, that tells you what stack you're on and when it stops being supported. And this year there's something new. 26.04 adds an HWE virtualization stack. qemu, libvirt, seabios, edk2, all in HWE variants, switched through a helper called ubuntu virt helper. Those update twice a year alongside the HWE kernel.



Corn

So the virtualisation layer gets the same treatment as the kernel now.



Herman

Same treatment, same cadence. Which matters if you're running virtual machines or containers on a desktop and you want newer QEMU features without moving off LTS.



Corn

There's one more mechanism. The stable release updates.



Herman

SRUs. This is the piece that's most relevant to the AI question later, so it's worth getting right. Normally, once an LTS is released, the versions of software in it are frozen. You get bug fixes and security patches, but not new upstream versions. SRUs are the exception. They're a defined process for pushing a newer version of a fast-moving piece of software into an existing LTS, without asking users to upgrade the whole distribution.



Corn

What qualifies?



Herman

Things where backporting a fix isn't practical because the fix depends on upstream changes, or where the upstream project only supports recent versions. Canonical uses it explicitly for ROCm, which we'll come back to, and for the container stack, containerd and runc. Those get newer versions delivered into the LTS as SRUs rather than waiting for the next release.



Corn

So there's a mechanism for admitting that some software can't sit still for two years.



Herman

There is. And the existence of that mechanism is the honest answer to a big chunk of Daniel's question. The LTS model isn't a flat freeze. It's a freeze with defined escape hatches, and the escape hatches are used for exactly the categories of software that can't tolerate a freeze.



Corn

Alright, so that's the maintenance model. The support window is longer than anyone needs, the point releases keep the ISOs current, the HWE stack quietly gives you a newer kernel on a stable release, and the SRU process exists for software that can't be frozen. Now the actual trade-off. What do you give up?



Herman

Less than the reputation suggests. Look at what 26.04 ships at launch. Firefox 150. LibreOffice 25.8, up from 24.2 in the previous LTS. Thunderbird 140. GIMP 3.2. GNOME 50. Python 3.14. Rust 1.93. LLVM 21. Go 1.25. GCC 15.2. PostgreSQL 18. PHP 8.5. MySQL 8.4 LTS. Valkey 9.0.



Corn

That's not a conservative stack. That's a current stack.



Herman

That's the thing. The "LTS means stale" line was true fifteen years ago when the gap between an LTS and upstream was measured in years. Now the gap at launch is months, and for a lot of these packages it's closer to nothing. Python 3.14 was current when 26.04 shipped.



Corn

Where does the staleness actually show up?



Herman

In the fast-moving stuff. And the first category is the desktop stack. 26.04 fully embraces Wayland. GNOME Shell can't run as an X.org session anymore. XWayland is still there so your old X applications keep working, but the compositor is Wayland only now. That's a real change.



Corn

And NVIDIA?



Herman

Fully supports Wayland now. That was the last holdout. For years the reason to stay on X was the proprietary NVIDIA driver, and that reason is gone.



Corn

So the base OS is current. What about applications that move faster than a two-year cycle, or faster than five years?



Herman

That's where Snaps, Flatpaks, containers and third-party repositories come in. And this is the part of the argument that has changed the calculus. Canonical's own framing for Snaps is that they're ideal for applications and tools that update frequently and independently from the main system. That's not marketing spin, that's an accurate



Herman (continued)

description of the isolation model. A snap carries its own dependencies and updates on its own schedule. The base OS doesn't care.



Corn

And the App Center now shows both.



Herman

Debs and snaps side by side. You see the same application, you pick which packaging you want. A snap can be confined, which means it's sandboxed away from the rest of your system, or classic, which means it has broader access and behaves more like a traditional package. Flatpak does the same thing from the other direction. Containers go further still.



Corn

A container is the most aggressive version of this. You give the application its own filesystem and its own libraries and it never touches yours.



Herman

Which is why it's the modern answer to "the LTS is too old for this." If a tool needs a library version your distribution doesn't ship, you don't upgrade the distribution. You put the tool in a box with the library it wants. Distro and application decouple completely.



Corn

So the honest summary of the freshness trade-off is that you give up very little at the base layer and nothing at the application layer.



Herman

Almost nothing. With one caveat, and it's a real one. The isolation model solves version conflicts. It does not solve driver and kernel-level questions. You can containerise an application's libraries. You can't containerise the kernel.



Corn

Which is where the AMD GPU question comes in.



Herman

And this is the case study where the whole episode gets tested, because GPU compute sits closer to the metal than anything else a desktop user touches. It's the place where the LTS-versus-freshness argument has the most teeth.



Corn

Set it up. What changed with ROCm in 26.04?



Herman

The big change is that ROCm is native in the Ubuntu archive now. You install it with apt. You install the rocm package or rocm-dev, and from that point on, apt upgrade manages it. Updates and security patches come through the normal package management path. Before this, if you wanted ROCm on Ubuntu, you ran AMD's amdgpu-install script manually, and you managed the versions yourself, and every kernel update risked breaking it.



Corn

So that's a genuine improvement. The thing that used to be the hardest part of running AMD compute on Linux is now a package install.



Herman

It is. And then the catch. 26.04 shipped ROCm 7.1.0. Upstream, at the time, was at 7.2 stable with 7.13 in tech preview. So the version in the archive was months behind. Phoronix put it bluntly: what shipped in 26.04 LTS was already months out of date compared to upstream.



Corn

Months out of date, in the one stack that moves fastest.



Herman

And Canonical's response is that the version it shipped establishes a foundation, and future updates get easier and faster, and newer versions will come through SRUs. So the plan is that 7.2 gets packaged and delivered into 26.04 without you upgrading the distribution.



Corn

Does the plan work?



Herman

That's the open question and I don't think anyone knows yet. Because Canonical also said something important about why it can't just track upstream automatically. Their words: with the pace of AI software development, sometimes breaking changes are introduced even in minor version changes. And those risk breaking applications built for older versions of ROCm.



Corn

So the reason the version lags isn't laziness. It's that moving the version forward can break the thing you built on top of it.



Herman

And it gets worse further out. ROCm 7.13 and the 8 series, the one they're calling The Rock, involves major library restructuring. Canonical has said it may not be able to do in-place upgrades that far. So the SRU mechanism, which works well for containerd and runc, has a harder job here, because the upstream changes are structural rather than incremental.



Corn

Which means the version gap might not close.



Herman

Might not close via SRU. And if it doesn't, the AMD GPU user has a decision the rest of us don't have.



Corn

What does AMD actually support? Because there's a wrinkle here that matters for anyone on an LTS.



Herman

This is the interesting part. AMD's own ROCm documentation for the 7.2.1 release lists Ubuntu 24.04.4 and 22.04.5 as the supported platforms. Not the interim releases. The LTS releases are AMD's primary supported targets.



Corn

So the vendor is targeting LTS.



Herman

The vendor is targeting LTS. And for consumer Radeon and Ryzen GPUs, the RX 7000 and 9000 series, those two LTS releases were the only supported platforms in that document. Then the 10.1.0 release notes add Ubuntu 26.04.1 with the 7.0 general availability kernel for Instinct, Radeon and Ryzen, and 24.04.5 on the 7.0 HWE kernel for Radeon and Ryzen.



Corn

So support arrives at the LTS releases, and it arrives at the point releases within them.



Herman

Consistently. Which is the opposite of what you'd assume. You'd think the fast-moving compute stack would chase the newest distribution. It doesn't. It targets the LTS releases, because that's where the install base is and that's what enterprise customers run.



Corn

So the "you need interim releases for GPU compute" argument is backwards.



Herman

For AMD on Ubuntu, largely yes.



Corn

What about the tooling layer? Ollama, llama.cpp, that world.



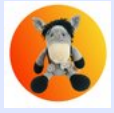
Herman

There's a package worth mentioning there. Lemonade Server. It's an open-source, OpenAI-compatible inference backend, and Canonical packages it as both a snap and a deb. It has out-of-the-box AMD support, including NPU support on Strix Point and Strix Halo, the Ryzen AI Max chips with up to a hundred and twenty-eight gigabytes of shared memory.



Corn

A hundred and twenty-eight gigabytes shared with the GPU.



Herman

Which is the pitch for those chips. You can hold a large model in memory on a laptop. And Canonical's framing for Lemonade is that it's similar to Ollama in many ways, but with out-of-the-box support for AMD hardware. That's the tell. The packaging exists specifically because the AMD path used to be the awkward one.



Corn

And it ships as a snap partly to dodge the library conflict problem.



Herman

Explicitly. The snap packaging is there to eliminate version and library conflicts. So the container-and-snap isolation model and the AMD GPU problem meet in the same place. The answer to "my ROCm version doesn't match what this tool wants" is increasingly "put the tool in a box."



Corn

And that gets us back to the central question. Does staying on LTS create a real disadvantage for local AI on AMD?



Herman

Here I have to be honest about what I don't know. The version gap is documented. 7.1 in the archive versus 7.2 upstream. What I have not seen is any benchmark comparing inference performance on LTS against interim releases. Nobody has published a llama.cpp or Ollama benchmark that isolates the distribution as the variable.



Corn

So we can say the version lags but we can't say what it costs.



Herman

We can't say what it costs in performance. My read is that it's a version-lag problem rather than a capability problem. The libraries are there, the GPU is supported, the acceleration works. You're not missing a feature. You're running a slightly older release of a stack that will get updated.



Corn

And where it does bite, the workaround exists.



Herman

The workaround exists and it's mature. ROCm supports multi-version installs, so you can have 7.1 from the archive and a newer one side by side. Docker images for ROCm are well established. Lemonade ships as a snap for exactly this reason. The modern answer to "the LTS is too old" is isolation, not a distribution upgrade.



Corn

There's a historical note here. Daniel asked about this before, years back, when he had an AMD card and wanted to run speech recognition locally.



Herman

The situation then was that AMD on Linux for compute was a fight. You were running install scripts, you were pinning kernel versions, and every update was a chance to break your setup. That was the state of things.



Corn

And now it's apt install rocm.



Herman

Now it's apt install rocm and apt upgrade maintains it. That's the arc. It went from being the hardest part of the platform to being a package.



Corn

So if the mechanisms are there and the version gap is survivable, when does it actually make sense to leave LTS?



Herman

Canonical's own guidance is clear. Interim releases suit people prioritising speed and rapid feature testing. LTS is recommended for production environments, enterprises and long-term projects. That's the official line and it maps onto what most people should do.



Corn

What's the third-party read?



Herman

Similar. LTS for production servers, stable development machines, family desktops. Interim releases for testers and hobbyists who want the newest Mesa drivers and the newest kernel, with the caveat that you should budget time for the upgrade treadmill. Every six months, you're moving.



Corn

The upgrade treadmill is the cost nobody prices in. It's not the upgrade itself. It's the weekend after.



Herman

It's the weekend after, and it's the compounding. You upgrade in April, you fix three things. You upgrade in October, you fix three more. Over two years that's four upgrade weekends instead of one, and each one has its own small chance of breaking something you depend on.



Corn

And there's a constraint on the LTS path that people don't always know. You can't skip.



Herman

The upgrade path is sequential. 22.04 to 24.04 to 26.04. If you're on 22.04 and you want 26.04, you go through 24.04 first. There's no direct jump without a clean install.



Corn

So staying on LTS is a commitment to a rhythm as well. Every two years, you move.



Herman

Every two years, and the move is well-tested because it's the path most users take. But it's still a move.



Corn

What hardware case remains for the interim releases?



Herman

It's narrow now, and it's narrower than it was. The hardware argument used to be the strong one. New laptop, needs a new kernel, so you needed the newest Ubuntu. But HWE absorbed that. Desktop installs track HWE by default. The kernel from this year's LTS is running on machines that installed the previous LTS.



Corn

So the hardware argument is mostly answered.



Herman

Mostly. What's left is the edge. If you want the absolute newest Mesa drivers, the ones that just landed upstream, before they've been through the HWE backport process, you need an interim release. If you want a kernel feature that's in 6.17 but won't be backported to your HWE stack, you need an interim release. Those are real cases. They're just narrow.



Corn

The virtualisation stack going HWE removes another one.



Herman

It does. That was a genuine reason to move, if you wanted newer QEMU or libvirt features on a desktop. Now those update twice a year on the LTS through the HWE variants. The list of things that require an interim release keeps getting shorter.



Corn

Downstream effects. What happens if the ROCm SRU works?



Herman

If 7.2 lands cleanly in 26.04 through a stable release update, the case for chasing interim releases weakens further, because the one stack people would have moved for is now current enough on LTS. If the ABI restructuring in 8 blocks it, AMD GPU users get pushed toward the options we talked about. Containers, multi-version installs, or, for the impatient, an interim release.



Corn

The AMD question has two futures and we don't know which one we're in.



Herman

We don't. And that's unresolved. Canonical has committed to packaging 7.2. Whether they can go past it depends on how disruptive The Rock turns out to be.



Corn

Let's pull it together for Daniel's actual question. Is staying on LTS a meaningful technical compromise in 2026?



Herman

Here's where I've landed. The freshness gap has changed character. It used to be a capability gap. The LTS couldn't do things the newer release could. Now it's a version-lag gap. You can do everything. You're doing it on slightly older libraries in a couple of fast-moving stacks.



Corn

The lag is mostly in the AI stack.



Herman

Mostly in the AI stack, which is the one place where months matter. Everything else, the browser, the office suite, the desktop environment, the language runtimes, the databases, those are current at launch and stay current enough.



Corn

Which means the compromise is real but it's localised.



Herman

It's localised to one layer, and that layer has a workaround. Which is why I don't think "meaningful compromise" is the right frame anymore. It's a deliberate trade with a known cost and a known mitigation. That's not a compromise. That's a decision.



Corn

The quiet story in all of this is HWE.



Herman

It is. Desktop installs already track newer kernels by default. The HWE stack brought Linux 7.0 to 24.04. Virtualisation now follows the same model. A large part of the reason people used to move to interim releases has been absorbed into the LTS without anyone announcing it.



Corn

Daniel's instinct to stay put is well-founded.



Herman

His instinct is well-founded and the mechanisms back it up. Ubuntu Pro free on five machines for the fifteen-year window, the HWE stack for newer hardware, Snaps and Flatpaks and containers for newer applications, SRUs for the fast-moving stacks. Every one of those is a deliberate answer to the problem of running a stable base for a long time.



Corn

The one place I'd tell him to watch is ROCm.



Herman

The one place to watch is ROCm. If he's running local inference on an AMD card and a model requires a newer ROCm than the archive ships, the container is the path, not the distribution upgrade. And if the SRU cadence turns out to be slow, that's the cost of the LTS decision, and he should know it's there.



Herman

It's the software equivalent of expecting the road to be built before you drive on it. Everyone assumes the newest thing is the one that's supported.



Corn

What Hilbert's getting at, underneath it, is that the assumption is backwards. The LTS is what the vendors target, what Canonical maintains longest, and what the HWE stack keeps current on hardware. The version lag is real but it's narrow and it has a container-shaped exit.



Herman

The answer to Daniel is that LTS in 2026 is not a compromise you tolerate. It's a position you can hold, as long as you use the mechanisms rather than ignoring them. Turn on Pro. Check your HWE status. Put the fast-moving tools in boxes.



Corn

The mechanisms will keep being tested, because the AI stack is the one layer that won't sit still for two years, and that's where the LTS model meets its hardest case every single cycle.



Herman

Which is worth watching. If Canonical can hold the ROCm line through SRUs, the LTS model survives its hardest test. If it can't, AMD GPU users become the exception that proves the rule.



Corn

That's where I'd leave it.



Herman

Thanks as always to our producer, Hilbert Flumingtop.



Corn

This has been My Weird Prompts.



Herman

If you enjoyed this one, leave us a review wherever you get your podcasts. It helps other listeners find the show.



Corn

We'll be back soon.



Hilbert

I want to know whether that thing they described would let me do something very specific. I have a cousin who works at a company that does enterprise Linux deployments. He's been telling me for years that LTS is the only responsible choice. And then last weekend he spent the whole time trying to get some new AI thing working on his work laptop and ended up using a container because the ROCm version was wrong.



Corn

Even the enterprise guy hit it.



Hilbert

Even him. But I want to correct one thing. The problem isn't the distribution. The problem is that software developers assume everyone is on the newest of everything. I've been saying this since I had to help my neighbour set up a printer. That was two thousand nine, and the assumption was the same then.



Herman

What was the printer problem?



Hilbert

The driver assumed a version of CUPS that nobody had. Same shape as the ROCm thing. Different decade. Now. I ran a production server on an LTS release for eleven years without a single reboot.



Corn

Eleven years.



Hilbert

It was a Raspberry Pi running a weather station in my shed. It was for a bet with my brother-in-law about whether it would rain on a specific Tuesday in two thousand eighteen. The bet is still unresolved. The Pi's SD card corrupted and I refuse to check the



Hilbert (continued)

logs because that would be cheating.



Herman

So the bet stands.



Hilbert

The bet stands. I have a second Pi running the same setup as a backup. It's been offline since two thousand twenty because I moved and never plugged it back in. And being an anteater, the old style keyboard that came with it, my claws never sat right on the keys, so I did most of the setup over the serial console.



Corn

The backup is in a box somewhere.



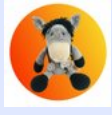
Hilbert

It's in a box. The bet is unresolved and it's going to stay unresolved. I'm not checking the logs on the card that corrupted because that's the whole point.



Corn

The cost of the LTS decision, Daniel, is that when the fast-moving layer gets too far ahead, you're putting things in boxes rather than upgrading the machine. That's the shape of it. The base stays put and the churn moves to the edges.



Herman

Hilbert's cousin is the proof. The enterprise guy, the one who says LTS is the only responsible choice, still ended up in a container last weekend. That's not a failure of the model. That's the model working as designed. The container is the escape hatch, and it does its job.



Corn

The thing I'll be watching is whether the SRU cadence for ROCm holds. If Canonical can move 7.2 into 26.04 cleanly, the LTS argument gets stronger every cycle. If the ABI changes in the eight series block it, the AMD users drift toward containers or interim releases, and that becomes the exception.



Herman

Either way, the direction of travel is clear. The base operating system is getting more stable and less relevant at the same time, because more and more of what you actually run lives outside it. That's not a problem for LTS. It's the environment LTS was built for.



Corn

Thanks to Hilbert Flumingtop, our producer.



Herman

If this was your kind of episode, go back for episode four, If Your Voice Ages, Does Your Fine-Tune Become Useless; episode five, Fine-Tuning ASR For Maximal Usability; and episode twenty-five, The Open-Source Battle for AI's Hardware Soul. This has been My Weird Prompts.



Corn

If you got something out of this one, a review on your podcast platform of choice helps other listeners find us.



Herman

If you've got a prompt of your own, send it to us on Telegram at t dot me slash MWP listener bot.



Corn

We'll be back soon.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **Canonical, release cycle & support matrix [primary]**
   https://ubuntu.com/about/release-cycle

2. **Canonical, kernel lifecycle & HWE [primary]**
   https://ubuntu.com/kernel/lifecycle

3. **Ubuntu 26.04 LTS release notes [primary]**
   https://documentation.ubuntu.com/release-notes/26.04/summary-for-lts-users/

4. **Canonical, 2026-09-29 [primary]**
   https://ubuntu.com/blog/upgrade-your-desktop-ubuntu-26-04-lts

5. **Canonical, Kevin Cazabon, 2026-06-03 [primary]**
   https://ubuntu.com/blog/amd-rocm-on-ubuntu

6. **AMD, 2026-04-14 [primary]**
   https://rocm.docs.amd.com/projects/install-on-linux/en/docs-7.2.1/reference/system-requirements.html

7. **AMD ROCm 10.1.0 release notes [primary]**
   https://rocmdocs.amd.com/en/develop/about/release-notes.html

8. **Canonical press release [primary]**
   https://canonical.com/blog/canonical-releases-ubuntu-26-04-lts-resolute-raccoon

9. **Phoronix, 2026-06-03**
   https://www.phoronix.com/news/Ubuntu-Newer-ROCm-SRUs

10. **Phoronix, 2026-04-23**
    https://www.phoronix.com/news/Ubuntu-26.04-LTS

11. **Ubuntu Portal, 2026-03-05**
    https://ubuntuportal.com/releases/ubuntu-releases-explained-2026/

12. **tux.fan, 2026-09-14**
    https://tux.fan/2026/09/14/ubuntu-26-10-vs-26-04-lts/