Episode Book

EPISODE #5782

Unraid in a VM: When the License Is the Wall

PUBLISHED	8 October 2026 at 11:50 UTC
DURATION	24:23
LISTEN	myweirdprompts.com/episode/unraid-vm-license-lock-in/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.2
BOOK GENERATED	19 October 2026 at 17:39 UTC

SYNOPSIS

Daniel runs a Synology for backups, but DSM fights his custom Python scripts — and two RAM upgrades left the box refusing to boot. His plan: keep the storage, attach it to his Ubuntu home server, and get the two things he actually values — extensible storage and discrete RAID. So can Unraid run as a VM on Ubuntu and still deliver software-defined parity with incremental expansion? The answer is yes, in three documented ways, and every one of them comes with a warning not to do it. This episode unpacks what an NAS actually is once you strip away the chassis, why Unraid isn't a RAID layer at all but an entire Slackware-based OS, the real cost of single-disk parity writes, and the licensing wall that turns a migration away from lock-in into a migration toward a different lock-in. Then we look at what Linux already offers — ZFS, mergerfs, SnapRAID, Btrfs — and which one actually fits a backup target that grows a drive at a time.

TRANSCRIPT



Corn

Here's the question that stopped me this morning. Is the thing you actually love about your NAS the NAS, or the RAID? Because those are two different purchases, and most people only find out which one they were paying for when they try to leave.



Herman

And that's Daniel's question, roughly, wrapped in a Synology and a bad RAM upgrade.



Corn

Roughly. Daniel's running a Synology for backup jobs, a few cheap drives in it. It's been fine. But he's grown out of it. The backup runners he actually cares about are custom Python scripts, and DSM fights him on running them. He's also been bitten twice by RAM upgrades that should have worked and left the box refusing to boot, which he reads, not unreasonably, as vendor lock-in wearing a compatibility badge.



Herman

Two RAM sticks that bricked a NAS. That's a real grudge.



Corn

It's a real grudge for a man in the middle of a migration. So his long-term plan is: keep the storage, attach it to the Ubuntu home server he already runs, and get the two things he identifies as the actual value of an NAS. Extensible storage he can add drives to, and its own discrete RAID. His words. "Which are two separate features from a storage perspective." He'd already done half our job before we started.



Herman

He had.



Corn

Then the actual question. Can Unraid run as a VM on his Ubuntu hypervisor and still deliver software-defined RAID with incremental expansion? And if that's not a good model, what's the better fit on top of Linux?



Herman

Let's start by asking what an NAS actually is, because he already told us.



Corn

He did. And the answer is not a box.



Herman

It's a bundle. And the bundle matters, because Daniel named the two parts himself. Parity-protected storage across many drives. And the ability to grow that storage incrementally without starting over. Everything else a NAS does, the Docker containers, the virtual machines, the web interface, the file browser, all of that is a separate product that happens to ship in the same chassis.



Corn

Which means the question "what's the advantage of a NAS over a server" has a slightly embarrassing answer for anyone already running a server. The advantages were two storage properties.



Herman

And one more thing, which is the strangest part of his question. Unraid is not a RAID layer. That's the trap in the phrasing. He's asking about running Unraid as a VM the way you'd ask about running a driver in a container. But Unraid isn't a driver. It's a whole Slackware-based operating system. It boots, it has its own kernel, its own userland, its own web management stack.



Corn

So "run Unraid as a VM" means virtualizing an entire operating system.



Herman

To get at one subsystem.



Corn

And in this case a subsystem the host operating system can already do.



Herman

That's the thing. If you want the array, and you already run Ubuntu, virtualizing a second NAS operating system inside Ubuntu is a strange path. You're paying for the whole OS in overhead and licensing to reach one feature. And the licensing turns out to be the wall, not the technology.



Corn

Before we get to the wall, settle the first half properly. What does Unraid's array actually do that plain Linux doesn't? Because that's the honest case for it, and the honest case is good.



Herman

The honest case is very good. Unraid's array uses a dedicated parity disk, and with dual parity, a second one. Parity one is XOR even-parity. Parity two is Galois-field Q-parity, Reed-Solomon territory. Functionally comparable to RAID six.



Corn

Two parity disks, tolerate two failures.



Herman

Exactly that. And here's the property that makes it feel different from everything else. Unraid does not stripe data. Each disk keeps its own filesystem. XFS by default, or ZFS or Btrfs if you choose. Nothing is spread across the set.



Corn

So any single disk stays readable.



Herman

Pull any disk out, plug it into any Linux machine, and you can just read the files. No array, no metadata, no reconstruction. It's a disk with files on it.



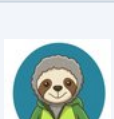
Corn

That's not just a technical nicety. That's the thing Daniel can't see in his Synology right now, and the thing he'd lose if he moved to a striped pool.



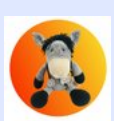
Herman

And the second property, the extensibility, is the hard one. Adding a data disk to Unraid requires one condition. The new disk has to be equal to or smaller than the parity disk. Same size or smaller, that's it.



Corn

That's a low bar.



Herman

Unraid clears the disk in the background, zeros it, while the array stays online. Then it formats it and adds it. No pool rebuild. No copying data off and back on. This is the promise that historically only Synology's hybrid RAID and Unraid did well, and it's exactly what Daniel said he wanted.



Corn

Here's the cost. There's always a cost.



Herman

There is, and it's the parity write penalty. Every write to a parity-protected array in read-modify-write mode is four disk operations. Read the existing data, read the corresponding parity, write the new data, write the new parity. Every single write.



Corn

And the array goes at the speed of the slowest drive in it.



Herman

Which is the killer detail. Unraid's own storage documentation puts read-modify-write at roughly twenty to forty megabytes a second typical write speed. In the best case, the fast drive can't help you, because the array waits for the slow one.



Corn

Twenty to forty.



Herman

For a pile of inexpensive drives from different years, that's realistic and it's not a bug, it's the arithmetic of single-disk parity.



Corn

And for Daniel's workload, that's probably fine.



Herman

For backup jobs, it's fine. Backup targets are mostly write-once, large sequential files. Nobody's running a database on it. But if he were planning to put virtual machine disks on the same array, he'd hate it within a week. Twenty to forty megabytes a second for a VM datastore is miserable. So the model is good for what he says he's doing, and bad for what he might drift into.



Corn

Which is worth saying plainly. He should decide now what the array is for. Backup target, or live storage.



Herman

Different answers.



Corn

Now the wall. He asked if Unraid can run as a VM, and the answer is yes in the sense that it boots. Unraid documents it themselves.



Herman

They have a dedicated page, and the page is remarkable for how unenthusiastic it is. Quote: "Lime Technology does not officially support this configuration for production data."



Corn

That's the vendor.



Herman

And then: "Virtualization introduces some overhead; expect reduced performance compared to running directly on hardware." And a third condition, which is the one that actually ends the conversation. Running it as a VM requires a separate, valid license key. You buy the license again for the virtual machine.



Corn

So the vendor publishes the procedure and says don't use it.



Herman

They publish the procedure because people do it, and they'd rather document it than have it happen blind. But read the framing. It's for testing new versions, developing plugins, kicking the tires on a release before you commit hardware. Not for the array holding your backup data.



Corn

Where does the license actually live? Because that's the part I want listeners to hear correctly.



Herman

The Unraid license is tied to the GUID of a physical USB flash drive. Not a file. Not a key you type in. A specific piece of hardware, a specific stick.



Corn

And a hypervisor has to give the guest a device.



Herman

So to run Unraid as a VM you pass a dedicated USB flash drive through to the virtual machine. And there's a further wrinkle. It has to come from a different manufacturer than the host's boot drive, or the VM won't see it. Same vendor, same identifiers, guest doesn't get it. Then the documented steps are renaming the flash drive's label, editing a config file on it, and creating a startup script to hand the device over before the guest boots.



Corn

That's a hack.



Herman

It's a hack with a vendor-published procedure, which is its own strange category. LinuxServer.io wrote a guide on this back in twenty fifteen. The community has kept it alive since. Multiple forum threads describe passing through not just the USB key but a whole PCI SATA controller, so the drives come through with it.



Corn

And then there's the extreme end.



Herman

The extreme end is the most revealing thing in all of this. A blog post from twenty twenty-three by a person called Giddi, who emulated the USB flash drive entirely in software.



Corn

No physical USB at all.



Herman

None. Using the Linux USB gadget API. Two kernel modules, one to present a dummy host controller, one to serve a mass storage device. Unraid booted. It even activated a trial license.



Corn

It worked.



Herman

It worked, and here's why I love the write-up. The author measured it. Unpatched, the gadget's read speed was about nine hundred and fifty kilobytes a second, which gave a boot time of five and a half minutes. He patched his kernel, got it to roughly seven and a half megabytes a second, and the boot dropped to about forty seconds.



Corn

Five and a half minutes to forty seconds.



Herman

And then he says, and I'm paraphrasing, that the dummy driver is intended for testing gadgets and debugging, not production. He knows what he built. He still published it.



Corn

Why? He says why.



Herman

His line is: "It was never about piracy, it was about convenience." And earlier, that he hates that Unraid requires a physical USB to boot, and that he prefers not to tie his entire infrastructure to specific hardware and the failure modes of a USB drive.



Corn

A man who spent a week emulating a USB stick so he'd never have to own one.



Herman

You can hear the whole personality in one sentence.



Corn

So the technology is not the wall. The technology works, in three different ways, all of them documented.



Herman

All of them documented by either the vendor or someone who warns you not to do it.



Corn

And the licensing is the wall, because what Daniel would be doing is leaving Synology over lock-in and buying a different lock-in. A license welded to a USB stick's identifier, plus a second license to run the first license virtually.



Herman

It's the same shape as the thing he's running from.



Corn

Down to the detail. He got burned by RAM that should have worked and didn't. The Unraid path hands him a vendor whose license checks a serial number on a piece of plastic.



Herman

That's an unflattering comparison and I think it's fair.



Corn

So the technology works and the licensing is the wall. Which raises the better question. If what Daniel wants is the array layer and not the operating system, what does Linux already have?



Herman

It has several, and they are not interchangeable. They differ on one axis that decides everything else, which is when parity is computed. Real-time or snapshot. And within the real-time ones, a second axis. Does the set self-heal, and can you mix whatever drives you already own.



Corn

ZFS first, because it's the default answer.



Herman

It's the default answer and the research complicates it. ZFS is mature, it's self-healing, it checksums every read. If a disk silently corrupts a block, ZFS knows and repairs it from parity. Nothing else in this conversation does that as well.



Corn

Discipline.



Herman

Real discipline. But its historical problem is rigidity, and the rigidity is precisely at Daniel's requirement. You grew a pool by adding whole vdevs, which means a whole second set of drives, or by replacing every drive in a vdev with a larger one, one at a time, and waiting for the rebuild each time.



Corn

So add a drive means add a shelf.



Herman

Add a shelf, yes. That has changed. RAIDZ expansion landed. Adding one disk at a time to a single-vdev RAIDZ1, RAIDZ2 or RAIDZ3 pool is real now, OpenZFS pull request fifteen thousand and twenty-two, and Unraid exposes that same feature from version seven point two upward.



Corn

And the limitations.



Herman

The limitations matter. You cannot change pool type. A mirror stays a mirror, RAIDZ1 stays RAIDZ1. You cannot promote it to RAIDZ2 later. Old data will not use the new disk's space until it's rewritten, so you get the capacity but not the redistribution. And the pool size is capped by the smallest disk in it.



Corn

That last one is the disqualifier for Daniel.



Herman

It is. ZFS wastes capacity with mismatched drives. A four terabyte and an eight terabyte in the same vdev loses you four terabytes. The eight behaves like a four. Daniel's entire premise is a few fairly cheap hard drives of varying sizes. ZFS wants a set of twins, and he's bringing a box of strays.



Corn

Btrfs second.



Herman

Btrfs is the opposite personality. Flexible block allocator, so it does incremental expansion, it does heterogeneous drives, and it can change RAID levels dynamically without stopping the array. In principle it's built for exactly his situation.



Corn

In principle.



Herman

RAID5 and RAID6 are labeled experimental. Unraid's own documentation calls them experimental, and the practitioner consensus is blunter than that. The phrase people use is that they cannot be relied upon. This is not a new caveat. It's been the standing state for years and it has not gone away, so parity across mismatched drives on Btrfs is a thing



Herman (continued)

you can configure and should not trust with the only copy.



Corn

Which leaves the union option.



Herman

mergerfs plus SnapRAID, which is the standard open-source answer for exactly this problem, mismatched drives you already own. mergerfs is a FUSE union filesystem. It presents many independent filesystems as one writable mount point. Every drive keeps its own filesystem and stays itself, and you see one folder.



Corn

And SnapRAID adds the parity.



Herman

SnapRAID adds scheduled, file-level parity. Up to six parity files, with per-file checksums, so it can tell you which file is wrong, not just that something is. And here is the whole trade-off in one word. Scheduled. The parity is only as current as the last sync, typically nightly.



Corn

So it's not protecting the write. It's protecting the state at two in the morning.



Herman

Which sounds strictly worse than Unraid, and for some workloads it is. If your data churns all day and a drive dies at four in the afternoon, you've lost a workday of writes on that disk. On the other hand, the overnight gap is a grace period. If you delete the wrong folder at noon, the parity file from last night still has it, and you can restore it. Unraid protects you the instant you write. SnapRAID protects you as of last night, which sometimes is what you actually wanted.



Corn

And for Daniel's stated workload?



Herman

Backup jobs. Write-once, large files, landing overnight. The parity is synced right after the writes land. The model fits him almost exactly. The inferior parity model is arguably the better one for his actual use.



Corn

I want that said once more, because it's the most useful idea in the hour. Real-time parity versus snapshot parity is not a quality ranking. It's a statement about what kind of data you have.



Herman

Right. Churny data wants real-time. Cold data wants snapshots. And the thing that decides which pooling system he should use is not which one is better, it's which one matches the write pattern he already has.



Corn

Now the migration half, because that's where his knees are actually shaking.



Herman

It's where the asymmetry lives. SnapRAID welcomes already-populated disks. You add the mountpoint, you run a sync, it builds parity over what's there. It doesn't need a blank drive and it doesn't care what's on it.



Corn

Unraid does the opposite.



Herman

Unraid requires new data disks to be cleared first. It zeros them before adding them, deliberately, so the parity is valid from the first moment. Which is correct engineering and means you cannot just hand it a disk you already filled.



Corn

And there's no path at all from his Synology into either one in place.



Herman

No. There's no adopt-the-array operation. Unraid's disks are XFS, Btrfs or ZFS, but the array metadata that ties them together isn't anything ZFS or mergerfs would recognize. Moving the disks into a different pooling system means copying the data out and putting it back, or reformatting.



Corn

Copying it out with what, if the machine doing the copying is the machine being rebuilt.



Herman

That's the practical problem and it's worth naming. You need somewhere for the data to sit while you rearrange. A staging disk, or borrow one, or accept an outage.



Corn

There's one consolation and I want it stated clearly, because it's the reason any of this is even conceivable.



Herman

Which is that Unraid, SnapRAID and mergerfs all keep disks independently readable. No striping. So if the array software dies, or you decide to move on, your data is never trapped behind a format. The files are on the drives. Worst case you plug them in one at a time and read them.



Corn

That's not a small property. That's the thing that makes the migration survivable at all.



Herman

Now, the closest thing to a drop-in. There is a project called nonraid, on GitHub under the account qvr.



Corn

What is it?



Herman

It's a fork of Unraid's storage array kernel driver. mergerfs's own documentation describes it as being like RAID5 at the block device layer without treating the collection of devices as a single device.



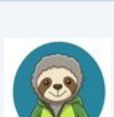
Corn

Say that again slowly.



Herman

Real-time parity across a set of drives, and each device stays individually accessible. Not one logical blob, a set of disks that happen to have parity between them. And you can combine it with mergerfs for the single mount point.



Corn

That is Unraid's array model, on a plain Linux host, without the license.



Herman

On paper, yes. I want to be honest here, though. I didn't fetch the repository directly for this. What I have is the description from mergerfs's own comparison page. So its maintenance status and how complete it actually is, I can't vouch for. It could be exactly what he wants or it could be half-finished. I don't know.



Corn

Which is itself a warning about relying on one project.



Herman

It is. And the last entry in the survey is a negative. There's a feature called ZFS AnyRAID, aimed at mixed-capacity disks with live redundancy and partial upgrades. That's the thing that would solve Daniel's exact problem.



Corn

And?



Herman

As of the middle of last year it was extremely early in development with no timelines. And there's no announcement since. So it is not a near-term option and nobody should plan around it.



Corn

The historical argument here is that only Unraid and Synology could do heterogeneous incremental growth. Where does that stand now?



Herman

It's weakening. RAIDZ expansion exists now, so ZFS can add a drive at a time. But ZFS still can't mix drive sizes efficiently and still can't change pool type. So the gap narrowed and did not close. They took one step toward Unraid and stopped.



Corn

Hm.



Herman

And that's the honest position for Daniel. If he wants Unraid's specific array behavior, real-time parity with mismatched drives and no striping, the options on plain Linux are nonraid, whose health I can't confirm, or running the real thing in the way the vendor tells you not to.



Hilbert

Anyway the tape is still the thing I'd point to. I never lost a drive that the tape didn't identify first.



Hilbert

I had a rack in a coat closet for a while. Nine drives at the peak, in a steel shelf unit that was not made for it, on a wooden floor that sagged about a centimeter in the middle. And I put a strip of masking tape on the front of every single drive with a Sharpie, the bay number, the size, the month it went in. That's all.



Corn

Just tape.



Hilbert

The tape is what saved me, not the software. When a drive went, I didn't have to power anything on to know which one it was. I didn't have to trust a management interface to tell me which bay the failure was in and hope the mapping was right. I walked in, read four strips of tape, pulled the one that said what I needed it to say. And then I had the old one in my hand and I could tell whether what was on it was still worth anything.



Corn

You still have the tape.



Hilbert

I have about two thirds of a roll. It's the good kind. It's a three-quarter-inch crepe masking tape, the beige stuff. You can't get it in that width here anymore, they only sell the thin painter's tape now, which is useless for a label because you can't read it from standing up.



Corn

That's the whole episode, isn't it.



Hilbert

Every one of these systems is an argument about whether you can read one disk by itself. Unraid, ZFS, Btrfs, all of them. You spent the hour on parity models. The question underneath all of them is whether you can pull a drive, hold it in your hand, and know what's on it. The tape answered that question better.



Corn

And that's exactly the property we kept circling.



Hilbert

Also I need you to bring the level up two clicks on this next one, you're clipping on the plosives.



Corn

Noted.



Herman

This is why his tape argument lands and our parity survey didn't. He's testing the drives with a labeling habit, and the notation is independent of the system that wrote it. That's the same property. Unraid, SnapRAID and mergerfs all pass it. Striped RAID doesn't, because a single drive out of a stripe is a fragment, not a disk.



Corn

He arrived at the thesis with stationery.



Herman

He did. It's a physical test for an architectural question. Can you read one disk alone, without the system that wrote it. And most of the systems we just discussed keep the answer yes.



Corn

Alright. That tape test is probably the cleanest way to think about everything we just went through.



Herman

And here's the part that should worry anyone planning a migration right now. The gap between Unraid and ZFS on heterogeneous incremental growth is narrowing, and it's not closed. RAIDZ expansion is real, but ZFS still can't mix drive sizes and still can't change pool type, and AnyRAID has no timeline. So the answer Daniel would get today may not be the answer in two years.



Corn

Which is a strange thing to plan around. If you build the pool this winter around the current limits, you're betting that the limit that pushed you toward one system stays the limit.



Herman

It might not. And the thing that protects you from guessing wrong is the one Hilbert was holding. Whether you can still read one disk by itself when the system that wrote it is gone.



Corn

That's the test. Not which parity model wins. What you can do with a single drive in your hand.



Herman

Which is the quiet irony of the whole thing, given Daniel's trying to leave one vendor's lock-in and the obvious replacement has its own, welded to a serial number on a USB stick.



Corn

A USB stick with a serial number is a smaller cage than a RAM compatibility list, but it's the same species of cage.



Herman

It is.



Corn

Thanks as always to our producer, Hilbert Flumingtop, who has a roll of tape with opinions.



Herman

This has been My Weird Prompts.



Corn

If you got something out of this one, leave us a review. It helps people find the show.



Herman

We'll be back soon.



Corn

See you then.