MY WEIRD PROMPTS

Podcast Transcript

EPISODE #418

RAID is Not a Backup: Mastering Home Server Resilience

Published February 02, 2026 • Runtime: 26:36

https://myweirdprompts.com/episode/home-server-data-resilience-snapshots/

EPISODE SYNOPSIS

In this episode of My Weird Prompts, Herman and Corn dive deep into the world of home server recovery after a listener's motherboard meltdown. They break down the crucial differences between hardware redundancy and data backups, exploring why file systems like BTRFS and ZFS are the ultimate tools for the modern self-hoster. The duo discusses the technical magic of Copy on Write (CoW) and how it allows for near-instant snapshots without eating up massive amounts of storage space. Whether you are building a "franken-server" with mismatched SSDs or seeking the enterprise-grade data integrity of ZFS, this episode provides a roadmap for making your data immortal. Learn about the "grandfather-father-son" rotation for automated backups and why bit rot is a silent killer you need to prepare for. It’s a masterclass in digital resilience, ensuring your next hardware failure is just a minor inconvenience rather than a total catastrophe.

DANIEL'S PROMPT

Daniel

I'd like to chat about backups and recovery for home servers. I've found that snapshots are the most practical tool for system recovery, especially when dealing with hardware failures. What would you recommend for someone who wants multi-disk support and robust snapshotting?

TRANSCRIPT

Corn

Hey everyone, welcome back to My Weird Prompts. I am Corn, and I am sitting here in our living room in Jerusalem. It is a bit of a chilly February afternoon, the rain is drumming against the window, and I am looking at my brother, Herman, who is currently surrounded by about three different technical manuals, a soldering iron that I hope is turned off, and what looks like a very small pile of SATA cables.

Herman

Herman Poppleberry, at your service. And yes, Corn, the cables are necessary. They are part of the aesthetic of a man who takes data integrity seriously. Although, to be fair, after hearing from Daniel this morning, I am feeling a little bit of that second-hand anxiety you get when a friend tells you their server just turned into a very expensive paperweight. It is February second, twenty twenty-six, and we are still dealing with the same fundamental hardware mortality that plagued us in the nineties.

Corn

It is that classic home server nightmare, right? You spend months or years tweaking your setup, getting all your media, your home automation, your personal files just right, and then one day you see the dreaded white smoke, or in Daniel's case, the motherboard just decides it has had enough of this mortal coil. He sent us a great prompt about his recovery process, but he had some very specific questions about how to make his next build even more resilient.

Herman

It is a rite of passage, really. If you have not spent a Saturday night with a headlamp on, peering into the guts of a mid-tower case while questioning every life choice that led you to self-hosting, are you even a nerd? But Daniel raised some really vital points in his prompt that I think a lot of people overlook until it is too late. Specifically, that distinction between RAID and backups, and the absolute magic that is snapshotting. He is running a bit of a franken-server—one NVMe boot drive and four mismatched SSDs—and he wants to know how to handle multi-disk support without losing the ability to roll back time.

Corn

Well, let us start there, because I think that is the biggest point of confusion for people just getting into this. Daniel mentioned that he learned the hard way that RAID is about continuity, not backup. Herman, can you break that down for us? Why do people get those two confused, and why is that mistake so dangerous?

Herman

It is the most common misconception in the storage world. RAID stands for Redundant Array of Independent Disks. The key word there is redundant. The goal of RAID is to keep the system running even if a physical hard drive dies. If one drive fails, the others pick up the slack, and you can keep watching your movies or accessing your files while you wait for a replacement drive to arrive in the mail. That is continuity. It is uptime. It is about the hardware surviving the hardware.

Corn

Right, so it protects you from a hardware failure of a specific component, the disk. But it does not protect the data itself from... well, us.

Herman

Exactly. Here is the catch: RAID does not protect you from yourself. If you accidentally type a command that deletes your entire photo library, RAID will happily and instantly delete that data across all your redundant disks with perfect efficiency. If a virus or a piece of ransomware encrypts your files, RAID will ensure that every single copy is beautifully encrypted. And as Daniel found out, if the motherboard or the power supply unit fails and sends a surge through the system, or if the file system itself gets corrupted due to a kernel bug, RAID is not going to save your data. It just means you have multiple copies of a broken system.

Corn

That is a great way to put it. RAID is a safety net for the hardware, but a backup is a time machine for the data. And Daniel specifically asked about snapshots as his primary recovery tool.

Herman

Precisely. And that is where Daniel’s focus on snapshots comes in. He mentioned that snapshots have been his most practical tool for recovery. And honestly, for a home server enthusiast in twenty twenty-six, snapshots are probably the closest thing we have to a superpower. They allow you to ignore the linear flow of time.

Corn

So, for the listeners who might be familiar with the term but have not actually implemented it, what is happening under the hood when we take a snapshot? Because it is not just a copy-paste of the whole drive, right? That would take forever and use up tons of space.

Herman

You are spot on. If you tried to copy two terabytes of data every hour as a backup, your server would do nothing but copy data all day. You would burn through the write endurance of those SSDs in months. Modern snapshotting, specifically in file systems like ZFS and BTRFS, uses something called Copy on Write, or CoW.

Corn

I love that acronym. It sounds so peaceful, but it is actually very high-tech. It makes me think of a cow just sitting in a field, calmly managing data blocks.

Herman

It is very efficient. In a traditional file system, when you change a file, the computer goes to the spot on the disk where that file lives and overwrites the old data with the new data. In a Copy on Write system, it never overwrites. Instead, it writes the new data to a fresh, empty spot on the disk. Only once that new data is safely written does the file system update its map to say, okay, the file now lives over here.

Corn

So the old data is still sitting there in its original spot?

Herman

Exactly. Until the system is told it can delete it, that old data remains untouched. A snapshot is essentially just a little note the file system makes that says, at ten o’clock on Tuesday morning, these were the locations of all the data on the disk. Because the old data is never overwritten, keeping that snapshot costs almost zero extra space initially. It only starts taking up space as you change or delete files, because the system has to keep the old versions around to satisfy the snapshot’s record of what the world looked like at ten o’clock. It is metadata magic.

Corn

That explains why Daniel was able to recover his virtual machines and his home assistant setup so easily. He just told the system to look at the map from before the crash. But he was asking about a specific challenge: multi-disk support and robust snapshotting. He mentioned he has an NVMe drive plus four different sized SSDs. That sounds like a bit of a headache for traditional RAID.

Herman

It is a total headache for traditional RAID. If you use a hardware RAID controller or even standard Linux software RAID, like MDADM, it usually wants all the drives to be the same size. If you have a five hundred gigabyte drive and a one terabyte drive, a traditional RAID one will just treat them both as five hundred gigabyte drives, and you waste half the space on the bigger one. It is inefficient and frustrating for home users who tend to buy drives whenever they are on sale. But Daniel is using BTRFS, and that is one of its biggest selling points.

Corn

Why is that? What makes BTRFS different when it comes to those mismatched drives? Does it just not care about the size?

Herman

BTRFS is much more flexible because it manages the storage at a chunk level rather than a disk level. It can look at a pile of different sized drives and say, okay, I will make sure that every piece of data has at least two copies on two different physical disks. It does not care if one disk is huge and the others are small, as long as it can find a place to put that second copy. For a home user who is constantly upgrading and might have a drawer full of old drives they want to put to use, BTRFS is incredibly forgiving. It is the ultimate recycler’s file system.

Corn

So Daniel is on the right track with BTRFS for his specific hardware setup. But he also mentioned ZFS. Now, Herman, I know you have a bit of a crush on ZFS. You have talked about it in the past as the gold standard. If Daniel were to switch, what would he be gaining, and what would he be giving up? Especially considering we are now in twenty twenty-six and ZFS has evolved.

Herman

Oh, the ZFS versus BTRFS debate. That is how you start a fight at a Linux convention. You are right, I do love ZFS. It was designed by Sun Microsystems for enterprise-grade servers, and it is incredibly robust. The main thing you gain with ZFS is data integrity. Every single block of data in ZFS is checksummed.

Corn

Checksummed. That means the system knows exactly what the data is supposed to look like, right? It is like a digital fingerprint for every file.

Herman

Right. It calculates a mathematical signature for every block. Every time it reads that block, it recalculates the signature and compares it to the original. If they do not match, it knows the data has been corrupted. This is what we call bit rot. Over time, cosmic rays or tiny electrical fluctuations can flip a single bit on a hard drive from a zero to a one. Most file systems will just read that corrupted bit and give you a broken file. ZFS will see the error, and if you have redundancy, it will automatically pull the correct data from the other mirror, fix the corrupted block, and then hand you the clean file without you even knowing there was a problem. It is self-healing storage.

Corn

That sounds incredible for long-term storage of things like family photos. But what is the catch? Why wouldn't Daniel just jump over to ZFS? Is it still as rigid as it used to be?

Herman

It is getting better, but it is still more rigid than BTRFS. ZFS organizes drives into groups called virtual devices, or vdevs. Historically, if you wanted to expand a ZFS pool, you had to add a whole new vdev. But, as of the last year or so, we finally have stable RAID-Z expansion in OpenZFS. This means Daniel could actually add a single disk to an existing RAID-Z group and expand the capacity. It was the holy grail of ZFS features for a decade, and we finally have it.

Corn

So the gap is closing?

Herman

A little bit. But ZFS still really prefers drives of the same size. If Daniel tried to put his four different SSDs into a ZFS RAID-Z configuration, it would likely default to the size of the smallest drive across the board. You lose that "mismatched drive" flexibility that BTRFS excels at. BTRFS is like the messy, creative artist who can make anything work with what is on hand, and ZFS is the disciplined engineer who demands everything be done by the book, even if the book is getting a few new chapters on flexibility.

Corn

That is a perfect analogy. For Daniel’s current setup, BTRFS is almost certainly the right choice. It gives him that multi-disk flexibility he needs for his "franken-server" while providing the snapshotting capabilities that saved his skin yesterday. But let us talk about the "robust" part of his question. He wants robust snapshotting. Just having the capability is one thing, but how do you make it a reliable recovery system? Because if you are manually taking snapshots whenever you remember, you are going to forget. And then when the motherboard dies, your last snapshot is from three months ago.

Herman

Exactly. Automation is the key to robustness. If it is not automated, it does not exist. For BTRFS, there are some great tools like BTRBK or Snapper. These tools can be configured to take snapshots every hour, every day, every week, and then automatically prune the old ones so you do not run out of space.

Corn

How does the pruning work? Do you just keep the last ten snapshots? That seems like it wouldn't give you much history.

Herman

Usually, you use a grandfather-father-son rotation. You might keep twenty-four hourly snapshots, seven daily ones, four weekly ones, and twelve monthly ones. That way, if you realize today that you deleted a file yesterday, you have an hourly snapshot. If you realize you deleted a file three weeks ago, you still have a monthly snapshot you can go back to. It gives you a very granular time machine without using a massive amount of disk space. And in twenty twenty-six, we also talk about immutability. You can set these snapshots to be read-only, which is a massive defense against ransomware. Even if a virus gets root access to your server, it cannot delete a read-only snapshot without a very specific, separate set of permissions.

Corn

That makes sense. But there is one more layer to this, right? Daniel mentioned he recovered his virtual machines using snapshots. It sounds like he might be using a hypervisor?

Herman

He likely is. If he is running Proxmox or even just KVM on Ubuntu, snapshots become even more powerful. You are not just snapshoting a file; you are snapshoting the entire state of a computer. RAM, CPU state, disk—everything. If you mess up a configuration in a virtual machine, you click one button and you are back to exactly where you were ten seconds ago. It is the ultimate "undo" button for server administration.

Corn

But Herman, we have to address the elephant in the room. A snapshot is still on the same physical disks as your data. If the disks themselves die, the snapshots die with them.

Herman

Exactly. A snapshot is a great first line of defense against accidental deletion or software corruption, but it is not a true backup until that snapshot is sent to a different physical device, preferably in a different building.

Corn

So we are talking about the three-two-one rule. Or is it the three-two-one-one-zero rule now? I feel like the nerds keep adding numbers to that.

Herman

They do! The classic is three copies of your data, on two different types of media, with one copy off-site. The modern version adds one copy that is air-gapped or immutable, and zero errors after automated recovery testing. And this is where ZFS and BTRFS really shine compared to traditional tools like RSYNC. They have a feature called send and receive. You can take a snapshot of your local server and then send just the changes, the incremental bits, over the network to another machine.

Corn

So if I have a second, cheaper server at my parents' house, I can just send the tiny differences every night? I do not have to re-upload the whole ten terabytes?

Herman

Exactly. Since it only sends the blocks that changed, it is incredibly fast and efficient. You could back up a multi-terabyte server over a standard home internet connection in just a few minutes every night. That is how you get true robustness. You have your local snapshots for quick recovery when you mess up a configuration, and you have your remote snapshots for when the literal house burns down or, more likely, when a power surge fries every component in your case.

Corn

I think that is a really important distinction. Daniel was lucky that it was just his motherboard that failed. His disks were fine, so he could just plug them into a new system and roll back his snapshots. But if that motherboard failure had taken the disks with it, he would have been starting from zero if he didn't have those snapshots replicated elsewhere. Which brings us to the hardware itself. Daniel was wearing a headlamp, looking into the case. That implies a physical intervention. One thing people often overlook when building these home servers is the quality of the power supply.

Herman

Absolutely. We talk about disks and file systems all day, but the power supply is the heart of the system. If it fails ungracefully, it can send high voltage through the data lines of your SATA cables and literally fry the controller boards on your hard drives. If that happens, no amount of RAID or local snapshotting will save you.

Corn

So, step one for a robust home server is actually a high-quality power supply and maybe an Uninterruptible Power Supply, or UPS?

Herman

A UPS is non-negotiable if you care about your data. It gives the server enough time to shut down gracefully if the power goes out. Sudden power loss is one of the leading causes of file system corruption. Even with a robust system like ZFS or BTRFS, you do not want to pull the plug while it is in the middle of a complex write operation. A UPS also acts as a line conditioner, protecting you from those tiny brownouts and surges that wear down components over time.

Corn

Okay, so let us recap the advice for Daniel. He has got a mix of drives. He likes BTRFS for the flexibility. He needs robust snapshotting. We are saying: keep using BTRFS for that multi-disk pool, use an automated tool like BTRBK to handle the schedule, and for heaven's sake, start replicating those snapshots to another drive or another machine.

Herman

And if he really wants to level up, he could look into a tool called Sanoid and its companion Syncoid. They are technically designed for ZFS, but the philosophy is what matters. Sanoid handles the policy, like how many snapshots to keep and when to take them, and Syncoid handles the moving of those snapshots between machines. There are similar scripts for BTRFS that do the same thing. The goal is to make it so you never have to think about it. It just happens in the background, and once a week you get a little notification saying your backups are healthy. In twenty twenty-six, we should be aiming for zero-touch data protection.

Corn

You know, it is funny. We are talking about all this high-level technical stuff, but it really comes down to peace of mind. I remember when you first set up our home server here in Jerusalem. You were so nervous for the first week, checking the logs every hour. But now, because we have that automated snapshotting and off-site backup, we do not even think about it. If a drive fails tomorrow, it is an annoyance, not a tragedy.

Herman

That is the goal. You want to turn a data disaster into a hardware chore. Replacing a motherboard is a fun Saturday project if you know your data is safe. It is a nightmare if you think you have lost ten years of photos. And Daniel mentioned something else in his prompt that I thought was interesting. He said that breaking things is how you learn. He’s been using Linux for twenty years, and he still feels like he learns the most when something goes wrong.

Corn

That is the absolute truth of technology. You don't really understand how a file system works until you have to recover it from a failure. You don't understand how networking works until your DNS breaks and you have to figure out why. There is a certain kind of "fearless learning" that comes from having a good backup. If you know you can't truly lose anything, you are much more likely to poke around in the settings and try new things.

Herman

It is like having a safety harness when you are rock climbing. You can try the harder moves because you know the fall won't kill you. A good backup system isn't just about protection; it is about freedom. It is the freedom to experiment, to break things, and to grow as a hobbyist or a professional. So, in a weird way, we should thank Daniel’s motherboard for failing. It gave him a chance to test his systems, it gave us a great topic to talk about, and it probably gave him an excuse to buy some new hardware.

Corn

I am sure he is already looking at new motherboards as we speak. Maybe something with a few more SATA ports or even some extra M-dot-two slots for those NVMe drives.

Herman

Oh, you know he is. Once you start down the path of the home server, you are never truly "done." There is always one more drive to add, one more service to host, or one more backup layer to implement. But he should also consider the endurance of those SSDs. Since he is using a mix of drives, he should check their TBW—Total Bytes Written—rating. Snapshots are great, but if he is doing a lot of heavy writes, he might want to ensure his most important data is on the drive with the highest endurance.

Corn

That is a deep dive for another day, Herman! I think we have given Daniel and our listeners a lot to think about. From the difference between RAID and backups to the magic of Copy on Write snapshots, and the great BTRFS versus ZFS debate.

Herman

It is a deep rabbit hole, but a rewarding one. If I had to leave Daniel with one final piece of advice, it would be this: test your recovery. It is one thing to see the snapshots in your list; it is another thing to actually try mounting one and pulling a file out of it. Do a "fire drill" once every few months. Make sure you know exactly what buttons to click when the pressure is on. If you are using BTRFS, practice the process of mounting a subvolume and swapping it with your current root. It is simple, but you do not want to be googling the syntax while your server is in pieces.

Corn

That is a great point. You don't want the first time you ever try to restore a snapshot to be when you are panicking because your server is down and your family is asking why the internet-connected lightbulbs won't turn on.

Herman

Exactly. Calm-time practice makes for a fast-time recovery.

Corn

Well said. Well, I think that wraps up our deep dive into the world of home server backups. Daniel, thanks for sending in that prompt and for giving us an excuse to geek out about file systems for a while. We hope your new motherboard arrives soon and that the migration is as smooth as those BTRFS snapshots make it look.

Herman

And to all our listeners, thank you for tuning in to another episode of My Weird Prompts. We really value this community we have built together. It is great to know there are so many of you out there peering into computer cases with headlamps on a Saturday night.

Corn

We really do. And hey, if you have been enjoying the show and you find these deep dives helpful, we would really appreciate it if you could leave us a review on your podcast app or over on Spotify. It genuinely helps other people find the show, and we love reading your feedback. It keeps us motivated to keep digging into these weird and wonderful topics.

Herman

It really does. You can find us, as always, at myweirdprompts.com. We have a full archive of all our past episodes there, and a contact form if you want to be like Daniel and send us a prompt of your own. We are also on Spotify, so make sure to follow us there so you never miss an episode.

Corn

Alright, Herman, I think it is time to go see if we can find that parts bin and see what exactly is on that motherboard from two thousand ten. I think I saw it under the spare ethernet cables.

Herman

I am telling you, Corn, that DDR3 memory is going to be worth its weight in gold one day! Or at least, it will make a very nice keychain.

Corn

We will see about that. Thanks for listening, everyone. This has been My Weird Prompts. We will see you next time.

Herman

Until next time! Stay nerdy, and keep those backups running.