Episode Book

EPISODE #5772

Two Jobs, One Word: What Agent Sandboxes Really Do

PUBLISHED	8 October 2026 at 08:43 UTC
DURATION	27:23
LISTEN	myweirdprompts.com/episode/agent-sandbox-two-meanings/
HOSTS	Corn, Herman
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.2
BOOK GENERATED	19 October 2026 at 17:39 UTC

SYNOPSIS

Sandbox" now means two different things: a security boundary that fences an agent off from your host machine, and a lightweight cloud Linux workspace where the agent does its actual work. Almost every product does both, but the two instincts pull in opposite directions — one wants to seal, the other wants to be useful. Drawing on roughly forty code-execution sandbox products launched in a single year, plus the ephemeral-versus-persistent divide and the reproducibility problem of drifting base images, this episode maps the whole category. Along the way: why a laptop is a bad home for a long-horizon agent, what changes the moment an agent gets a real command line, and why "a sandbox is what ceases to exist when nobody's watching" might be the sharpest line in the debate.

TRANSCRIPT



Corn

Right off the top, because this one needs a map before we walk into it. Daniel wants to look at agent sandboxes, and his point is that the word has quietly split into two jobs.



Herman

Two jobs sharing one word. That's the whole episode.



Corn

First meaning, the one most of us reach for: security. Sandboxing as a boundary so the model's code can't get at the host machine or your secrets. Sometimes obtrusive, in his words, sometimes the restrictive kind that fences an agent into a local box and gets in the way.



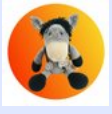
Herman

And the second meaning...



Corn

The second meaning, which he thinks wins: sandboxes as lightweight, often ephemeral filesystems for agents that live in the cloud. His argument is that the runtime is moving off your laptop. Desktop and mobile clients become thin viewports, they don't run the agentic process at all. And the moment an agent goes past text in, text out, it needs real Linux command line tools. Once it has those it can take an uploaded binary, edit audio, run a Python script.



Herman

Which changes what a sandbox is for.



Corn

Then he wants a survey of the popular approaches to provisioning these environments, and a clean line drawn between ephemeral and persistent remote environments. His example being a dev workspace, where you want a pinned Python and pinned dependencies that survive between sessions instead of being rebuilt every time. And from there, replicable base images, image libraries, the way cloud platforms hand you container images.



Herman

So we're using one word for two different products.



Corn

We are. Let's start with the fact of that, and then figure out when the two meanings stopped being able to ignore each other.



Herman

Cleanest way to split them. Meaning one, the security boundary. An isolated environment where the agent can run code freely and the host, the other tenants, and your credentials stay on the other side of the wall. That's the older meaning, the one that's been in computing since long before any of this.



Corn

Meaning two is a Linux workspace. A disk, a shell, a package manager, a filesystem the agent can actually work in. Cloudflare's version of that is nearly blunt about it, their line is essentially "your agent needs a computer."



Herman

And those two meanings aren't at war. Almost every product doing one is doing both.



Corn

They pull in different directions, though.



Herman

They do. A security boundary wants to be as small and as sealed as possible. A workspace wants to be useful, which means it wants tooling, network access, storage, all the things that make a boundary leaky. Same product, two instincts, and the vendors mostly answer whichever question they find more flattering. Buyers keep asking the other one.



Corn

Which is why the category feels foggy from outside.



Herman

And the category is enormous. Somebody on Hacker News sat down and counted roughly forty code execution sandbox products that launched in a single year. Forty. That's not a market finding its shape, that's a market where everyone smelled the same opportunity at once.



Corn

So the security meaning is the one we all know, the filesystem meaning is the one that got urgent, and to see why it got urgent you have to look at where the agent actually runs now.



Herman

Docker's framing is the sharpest version of this and they wrote it almost as a complaint. Agents do long horizon work now, tasks that run for hours, not minutes. And a laptop is built around a person. It sleeps when the lid closes, it throttles on battery, it drops the connection the second you walk out of the building with it.



Corn

The agent gets interrupted because you wanted lunch.



Herman

Because you wanted lunch. Cloudflare says the same thing from the demand side: agents create sandboxes on demand, per task, they expect them to be ready immediately, and then they expect to be able to pause and resume. That's not how a laptop works. That's not how anything on your desk works.



Corn

So the process moves to the cloud.



Herman

The process moves to the cloud, the phone becomes a viewport, and now the sandbox isn't a safety feature sitting beside the agent. The sandbox is the agent's computer. It needs a disk, a shell, and a way to install things.



Corn

Here's the part I find interesting, and it's the threshold Daniel points at. Text in, text out. As long as that's the only thing happening, the sandbox is basically a padding cell. The moment the agent gets a Linux command line, everything changes.



Herman

Concretely, that means it can clone a repository, install packages, configure the environment for the task it's been given. Cloudflare ships an image for exactly this, Debian Trixie Slim with Node twenty four twenty LTS baked in, and through their exec path the agent does all three of those things inside its own little machine.



Corn

There's a cleaner illustration in the Kubernetes agent sandbox work. The code execution use case there is a FastAPI endpoint. You post to it, it runs the code, it hands back stdout, stderr, and the exit code. That's it. But it's running inside a container with its own filesystem, its own processes, its own network stack.



Herman

That's the primitive. That's the whole thing in about three sentences.



Corn

And then the user facing payoff, which is where it stops sounding like infrastructure and starts sounding like a product. You upload a binary. The agent downloads it, runs it, and suddenly it's editing your audio file or executing your Python script or transcoding your video. The sandbox stopped being a fence and became a capability.



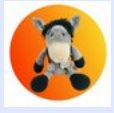
Herman

The fence never went away. It just stopped being the interesting part.



Corn

So ephemeral versus persistent. Give me the actual mechanical difference, not the marketing version.



Herman

Nirvana drew the clearest line I've seen on this. Ephemeral sandbox: no disk at all. Pure compute. It boots in about two and a half seconds. Persistent sandbox: it gets a dedicated volume, mounted at workspace, first boot takes six or seven seconds, and everything the agent writes into that folder survives a pause and survives a resume. Their rule of thumb is ephemeral for stateless jobs, persistent for long running agents that need to checkpoint and continue.



Corn

Two and a half seconds versus six or seven. That's the price of a disk.



Herman

And the disk is the entire difference.



Corn

Somebody's going to hear "persistent" and assume it means "never goes away." It doesn't.



Herman

It means the folder goes away slower. It means state you deliberately wrote survives the lifecycle event. It doesn't mean the machine is standing there waiting for you.



Corn

There's a counter argument to all of this that I want on the table, because it's the most interesting thing I read preparing for this.



Herman

Fly's computers versus sandboxes piece.



Corn

Their thesis is that persistence, isolation, wake latency, none of those are the real distinction. The real distinction is what happens when you stop paying attention to it. A sandbox is something that ceases to exist when nobody's watching. A computer is just there. You leave it, you come back, it's still there, it didn't decide anything while you were gone.



Herman

That's a strong challenge to the ephemeral orthodoxy, and I don't think it's just marketing. If you watched the last year of this category, the direction of travel is persistent. Vercel turned persistence on by default at GA. Fly built Sprites around a hundred gigabyte persistent disk. Docker's cloud sandboxes default to a one hour run that can stretch to twenty four.



Corn

So the ephemeral pitch and the persistent buildout are happening in the same twelve months.



Herman

Which usually means the ephemeral framing is the sales pitch and the persistent version is the product.



Corn

Now the piece Daniel's specific about, reproducibility. Why can't a dev workspace sandbox just be rebuilt from scratch every session? Everyone's building tools fast, containers boot in under a second, why does it matter?



Herman

Because you're not rebuilding the same thing. You're rebuilding something that looks the same and isn't. You pin Python to a specific minor version, you pin your dependency tree, and then six weeks later a rebuild quietly resolves a different transitive dependency, or the base image upstream moved, and now the environment your agent tested against is not the environment it's running in.



Corn

Small version drift.



Herman

Small version drift, and the failure it produces is the worst kind, which is a test that passes locally and fails in CI, or the reverse, and nobody can reproduce either one.



Corn

So the answer is snapshots and templates.



Herman

The answer is that you stop treating the environment as something you build and start treating it as something you copy. Cloudflare's snapshot model is the clean version, one snapshot starts many isolated environments from the same baseline, which cuts setup time and, their words, prevents environment drift. And they pitch it specifically for evals, comparing a prompt against a skill against a model, where if the baseline moves you've contaminated your comparison.



Corn

Modal does the same thing with snapshot filesystem, hands you back a reusable image. There's a demonstration where you prepare one state and fork three parallel sandboxes off it.



Herman

Which is a new capability. You couldn't do that with a laptop.



Corn

You could, you'd just have three laptops.



Herman

Kubernetes is doing it with custom resources, a sandbox template and a warm pool. The warm pool pre boots pods so that when a new environment gets requested it allocates fast instead of cold.



Corn

And Docker's version is the kit, which is a pre configured pre built sandbox for an agent, and they ship them for Claude Code, Codex, Copilot, Antigravity, Open Code, Hermes.



Herman

That's the image library Daniel was describing. Same idea as pulling a postgres image off a registry instead of writing a Dockerfile on a Tuesday night.



Corn

Except now the image isn't just the application's environment. It's the agent's entire sense of what a computer is.



Corn

So that's the fork. Ephemeral or persistent, and a base image either way. Which brings us to who's actually selling which version of it, and what the numbers say when you put them side by side.



Herman

Organize it by pattern rather than by name, because that's the only way it stays legible. First pattern: microVM per sandbox. E2B is the reference there, Firecracker underneath, boots around a hundred and fifty milliseconds, pause and resume preserving both memory and filesystem. And their infrastructure is Apache licensed, you can self host on Terraform and Nomad and Consul.



Corn

Open underneath.



Herman

Vercel is the same isolation primitive, Firecracker, but they went the other way on persistence and turned it on by default when it hit GA in May. Then in September they put Drives into public beta, durable volumes that outlive the sandbox itself. Fly's Sprites are



Herman (continued)

Firecracker too, but they're the purist end of persistent, hundred gigabyte ext4 disk, supervised services, a public HTTPS URL per Sprite, and credential connectors.



Corn

Second pattern?



Herman

Workspace centric. Daytona is the one there. Docker container by default, Kata or Sysbox if you want the stronger isolation, and the whole product is built around a persistent dev environment, with fork, pause, resume, and snapshot. It's the pattern that most closely matches what Daniel described, a workspace that stays stable between sessions.



Corn

Third?



Herman

Sandbox as one primitive inside something bigger. Modal is the example. gVisor for isolation, VM sandboxes in beta now, snapshot and fork supported but no in place resume. The sandbox is a feature of their serverless GPU platform rather than the product.



Corn

Fourth pattern is platform native, where the sandbox is an extension of something you already run on.



Herman

Cloudflare is the clearest, containers on Workers plus Durable Objects, with a durable object scheduling policy, snapshots, and programmable egress handlers that live outside the sandbox. Vercel sits here too for the platform half. Runloop runs a dual layer, microVM plus container, with suspend and resume, and they've built agent gateways and an MCP hub around an eval focused workflow. Northflank lets you pick your isolation per workload,



Herman (continued)

Kata or Firecracker or gVisor, ephemeral by default with optional volumes, GPU support, and bring your own cloud.



Corn

And then the last pattern, which is the one that isn't a vendor at all. Standardize the primitive.



Herman

Kubernetes SIG agent sandbox. Four custom resources, sandbox, sandbox template, sandbox claim, sandbox warm pool, gVisor or Kata for isolation, and integrations for the OpenAI Agents SDK, LangChain, OpenHands, DeepAgents, an MCP server, Gymnasium for reinforcement learning. That's an attempt to make the sandbox a boring cluster object instead of a product you buy.



Corn

Microsoft's version is the other half of that.



Herman

MXC, Microsoft Execution Containers, that's OS level containment rather than a cloud sandbox, four backends, process, session, WSL, microVM, and three modes, enforcement, learning, permissive. Their framing is the one worth carrying around: an agent cannot be its own security authority. The policy stays outside the agent's control so the agent can't grant itself more access.



Corn

Which is a sentence that applies to cloud sandboxes too, they just don't say it as plainly.



Herman

Now the numbers, and I want to handle these carefully because the headline numbers are the most misused thing in this entire category. ComputeSDK ran a burst test, bringing up sandboxes under load. Vercel, zero point six seven seconds median, one point one two at P ninety nine, and a hundred percent success rate. Modal, zero point eight eight. Runloop, zero point eight nine. E2B, one point six one. Cloudflare, five point zero six. Daytona, zero point two seven seconds median.



Corn

Fastest in the set.



Herman

At a thirty seven percent success rate.



Corn

...Say that again.



Herman

Thirty seven percent. Nearly two out of three requests didn't come up at all, and the ones that did were the fastest in the field. That is not a performance win, that's a benchmark methodology warning. A median calculated over a third of your attempts is telling you about your attempts, not your platform.



Corn

LogRocket measured cold starts and got a different order entirely. E2B at seven hundred seventeen milliseconds to create, six hundred sixty two to resume. Vercel eighteen fifty two to create, thirty three thirty three to resume, which is a very different number from zero point six seven.



Herman

Because it's a different measurement. One is a burst test under contention, one is a single cold start in isolation. Same vendor, one number is medians under load and the other is comfortable single shot latency, and the two aren't comparable. That's the trap the whole category falls into. You cannot line these up across sources and declare a winner.



Corn

What about vendors measuring themselves?



Herman

Cloudflare rebuilt their container stack in September and their numbers are a rebuild story, median startup from four point zero four nine seconds down to six hundred forty eight milliseconds, six point two times faster, P ninety nine from six point seven down to one point one. Then a burst test where they bring up a hundred thousand containers in five point three eight seven seconds across six locations.



Corn

Their own benchmark, their own hardware, their own definition of ready.



Herman

Entirely. Perplexity published SPACE and said median create latency went from a hundred eighty five milliseconds to sixty, three point one times, and P ninety from four forty seven down to eighty nine. And they claim millions of sandbox creations and tens of millions of reconnects in launch week.



Corn

Those are honest numbers presented by interested parties.



Herman

Which is the whole category, and it's fine, as long as you don't treat any of them as neutral.



Corn

What actually differentiates these things, then, if it isn't the cold start?



Herman

Billing model, and it's the most useful thing in the entire survey. Two camps. Wall clock billing, and active CPU billing. Vercel and Cloudflare bill active CPU. On an agent loop that's mostly waiting, think model inference, tool round trips, the sandbox talking to a server that's thinking, active CPU billing can come out roughly half the cost.



Corn

And the same platform becomes the most expensive option in a different workload.



Herman

The same platform becomes the most expensive option the moment you're running a stateful agent that just sits there resident between tasks. Because now the wall clock is running and you're not using CPU, so the billing advantage evaporates and then inverts. The cheapest provider is not a fixed answer. It flips depending on the shape of your workload, and almost nobody selling says that out loud.



Corn

Give me the raw rates.



Herman

E2B and Daytona both at five point zero four cents per vCPU hour, one point six two cents per GiB hour. Modal around seven point one cents and two point four. Vercel twelve point eight cents active CPU and two point one two for memory. Cloudflare seven point two active and less than a cent for memory. Fly Sprites seven and four point three seven five. Runloop ten point eight and two point five two. Northflank one point six seven and point



Herman (continued)

eight three, cheapest headline in the set. Docker's cloud sandboxes run from seven cents an hour for a single vCPU and two gig, up to a dollar twelve for sixteen vCPU and thirty two gig.



Corn

Now the traps.



Herman

The traps are where this gets nasty. Start with egress precedence, which is a footgun of the purest kind. E2B resolves allow over deny. Vercel resolves deny over allow.



Corn

So the same policy document means opposite things depending on which one you deploy it to.



Herman

Silently. No error, no warning. You write a policy with an allow rule and a deny rule that overlap, one platform lets the traffic through, the other blocks it, and both think they did what you asked. And E2B documents something worse: blocked TCP connections can look successful from inside the sandbox. The connection appears to work, from the agent's point of view, and it isn't going anywhere.



Corn

So the agent thinks it's talking to the network.



Herman

It thinks it's talking to the network.



Corn

What about isolation? Everyone claims it.



Herman

Isolation is a stack, not a checkbox, and here's the case that proves it. BeyondTrust looked at AWS AgentCore. Firecracker compute isolation held perfectly. The microVM did its job. But DNS egress leaked, and an over broad IAM role let the researchers read S3 buckets containing personal data. One stack layer held, and two others failed, and the report reads like a breach. AWS remediated it in April.



Corn

The strongest isolation primitive in the industry, and it didn't matter.



Herman

It didn't matter on its own. Ladder goes Firecracker microVM at the top, then gVisor, then shared kernel container, then a V8 isolate at the bottom. Choosing a rung is not the same as building a boundary.



Corn

You mentioned something about vendors not saying which rung they're on.



Herman

Daytona and Blaxel both don't disclose their isolation primitive in public documentation. Fly pointed this out. Daytona says complete isolation, a dedicated kernel, and never names what provides it. Blaxel says instant launching virtual machines and doesn't say which microVM. And Daytona's production codebase went closed source as of June, despite earlier positioning around AGPL and self hosting. The sources disagree on whether self host is still available.



Corn

That's a strange thing to be coy about.



Herman

It's a very strange thing to be coy about. You're selling isolation. Naming the primitive should be the easiest part of the pitch.



Corn

So what actually sets a good one apart, if it isn't boot time and it isn't the isolation label?



Herman

Credential brokering. That's the real differentiator, and it's the thing four platforms all landed on independently. Cloudflare, Vercel, Runloop, and Fly all inject credentials outside the sandbox, so the agent never holds the key. It makes an authenticated call to something that holds the secret for it.



Corn

So a prompt injection gets the agent to try something and there's nothing to steal.



Herman

There's nothing in the box worth stealing. Which is the structural answer to the oversight problem, and it's the answer nobody would have predicted five years ago, because the instinct then was to firewall the network instead. Fly's line about this is the best sentence in the category and I'm quoting it: you are handing live credentials to a process whose entire job is to run code that a language model wrote, and the security boundary is vibes.



Corn

That's an unkind sentence to the entire industry and I think it earns it.



Herman

They follow it up with a four part test. You need a disk, supervised processes, an address, and a credential broker, and if you don't do all four, you've built a sandbox with better marketing.



Corn

So that's the survey, and here's what I take from it. The category has converged on the primitive, a Linux workspace with a disk and a shell, but it has not converged on the contract.



Herman

Not remotely. Persistence semantics differ silently, egress precedence differs silently, isolation disclosure varies from complete to nothing, and the billing model determines which one is actually cheaper for you. There's no spec anyone agreed on.



Corn

And the things that look settled mostly aren't. Cloudflare's disk resets on sleep, a sleeping container comes back with a fresh disk from its image. Vercel turned persistence on by default, which is the opposite default. E2B's timeout behavior defaults to kill, not pause, so if you didn't explicitly ask for pause at creation time, unsaved work is gone.



Herman

Gone. Silently, on a timer, because a default you didn't know existed did the reasonable thing from the platform's point of view.



Corn

A minute ago you said something about the base image being the thing that's actually trusted, and I want to pull on that, because I don't think we've earned it yet.



Herman

Spin up a sandbox, it pulls Debian Trixie Slim, and you trust everything in it. You didn't build it, you didn't audit it, you didn't choose the person who built it. You chose a name and a tag.



Corn

And the app store is still cold.



Herman

...The pin is only as good as the base image underneath it.



Corn

Go on.



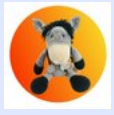
Herman

You pin Python three point twelve exactly, you pin your dependency tree, you write it all down, and none of it matters if the layer underneath shifted. I spent a stretch of my life doing nothing but building environments for other people's code and then building them again, and the thing that kept biting us was never the runtime. The runtime behaved. The image was the problem. I have seen a base image rebuilt from a different upstream than the one it claimed to be.



Corn

Same name, same tag.



Herman

Same name, same tag, different tree. Nobody noticed for weeks, because nothing failed loudly. Things just resolved slightly differently and a test that used to pass stopped passing, on a machine that supposedly hadn't changed.



Corn

And you're saying that's still true at scale.



Herman

I'm saying nobody audits the chain. The survey we've just done is all vendor side and benchmark side, cold starts, pricing, isolation tiers. Not one of those numbers tells you whether the image you're starting from is the image it says it is. That's the layer everyone skips, and it's the layer everything above it stands on.



Corn

So the sandbox is only as reproducible as the thing it was cloned from.



Herman

And nobody in the category is selling you that.



Corn

Which is uncomfortable, given that the snapshot model is pitched on exactly the promise of preventing environment drift.



Herman

Right, but a snapshot is only as trustworthy as the baseline you snapshotted. Cloudflare starts many environments from the same baseline, that's real, that's a genuine improvement. But the baseline is an image somebody else assembled, and the drift problem didn't get solved, it got pushed up one level and hidden behind a nicer interface.



Corn

The drift moved into the basement and took the sign down.



Herman

Exactly that.



Corn

What do you want to leave people with, then?



Herman

That the primitive has settled and the vendor crowd hasn't. If you're choosing today, the headline cold start number is the least useful number on the page. What matters is the billing model against your workload shape, whether credentials are brokered outside the boundary, how egress precedence resolves, and what the disk does when nobody's watching. Those four questions separate the real products from the marketing.



Corn

And the second order implication is the one I keep circling back to. If the sandbox is the agent's computer, then the base image is the agent's operating system. And nobody's auditing the operating system.



Herman

Nobody's auditing the operating system.



Corn

The standardize the primitive moves, Kubernetes, Microsoft, are the counter pressure to that. Watch whether the contract gets written down before the category consolidates. Because once it consolidates, whoever won gets to write the semantics, and right now those semantics are whatever each vendor felt like doing on the day they shipped.



Herman

Most common wrong belief about all of this, if I had to name one?



Corn

That sandbox means one thing. People hear it and assume security, or assume workspace, and then half the conversation is two parties answering different questions in good faith and getting angry about it.



Herman

Security boundary or the agent's computer. One word, two products, and the vendors mostly don't help you tell which one you're being sold.



Corn

That's the episode. Thanks to Hilbert Flumingtop, our producer.



Herman

This has been My Weird Prompts. If you've got a minute, a review helps more than you'd think.



Corn

You can find us at my weird prompts dot com. We'll be back soon.