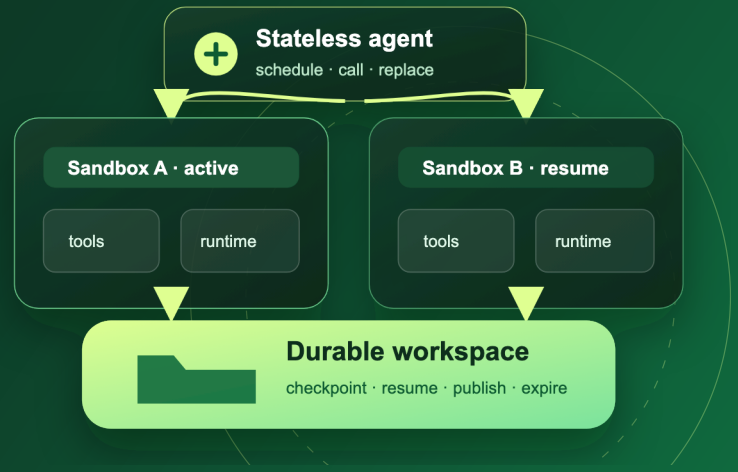
AGENT INFRASTRUCTURE · DURABLE WORKSPACES

Stateful Sandboxes for Stateless Agents

Designing Durable Volumes for MCP Tool Workspaces

Rui Su co-founder of Juicedata



ONE TASK · FIVE TOOL CALLS

The agent is stateless. The work is not.

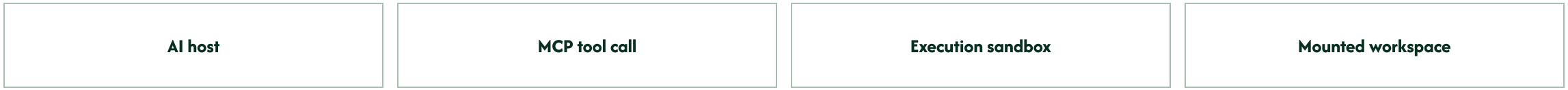
01 · checkout	Repo baseline and downloaded inputs enter the sandbox.
02 · edit	Uncommitted changes create unique mutable task state.
03 · install	Dependencies grow a useful—but rebuildable—cache.
04 · test	Partial evidence and logs accumulate in the workspace.
05 · report	The result is not durable until it is explicitly published.

Sandbox lost after test

Five calls created one evolving filesystem.

What must survive—and what should disappear?

MCP defines the call—not the workspace behind it



The call path is standardized; workspace semantics are not.

What MCP gives you

Discovery	Find available tools.
Invocation	Call a named capability.
Result	Return structured output.

What the platform must still define

- **Mount**
- **Sharing**
- **Persistence**
- **Recovery**

Workspace: filesystem state visible and mutable across tool calls

MECHANISM IS NOT INTENT

A persistent volume can preserve the wrong things

VOLUME ASKS

Can I preserve these bytes?

Persistence · consistency · snapshots · replication · sharing

POLICY ASKS

Should they survive—and under whose control?

Value · owner · boundary · retention · publication · deletion

Durability is a policy, not a volume feature. A volume can retain a secret too long and still lose an edit at the wrong commit boundary.

SIX STATE CLASSES

Not all agent state deserves the same durability

Workspace-adjacent

Files the task touches

Inputs	Versioned · reproducible · read-only
Mutable work	Resume or review after failure
Cache	Rebuildable · short TTL · loss acceptable

Systems of record

State belongs elsewhere

Artifacts	Immutable ID · explicit publication
Control	External · ordered · auditable
Secrets	Runtime-only · never snapshot

PLACE EACH KIND OF STATE DELIBERATELY

The workspace is one state plane, not the whole agent

Read-only inputs	Control DB	Secret broker	Artifacts
Git · OCI · objects	status · lineage	short-lived only	stable ID · retention

Workspace volume · code, files, logs, large payloads

Filesystem data plane

- read
- write
- rename
- fsync
- shell tools

Explicit control / action plane

- checkpoint
- publish
- approve
- deploy
- delete

Interface: POSIX · files Skill · runbook MCP · capability Admin API · high-risk action

Filesystem plane and control plane

FROM RUNTIME FAILURE TO INTENTIONAL DELETION

Durability is a contract across boundaries

01 · Runtime

Process crash

Which operation commits a write?

write · close · fsync

02 · Sandbox

Restart / reschedule

How much acknowledged work may be lost?

RPO

03 · Infrastructure

Node or zone loss

How quickly can another sandbox resume?

RTO · remount · failover

04 · Lifecycle

Retain → delete

Who may restore, fork, publish, or erase?

Retention · authority

Contract: commit point RPO RTO retention authority

Persistence becomes testable semantics

EVERY TRANSITION IS AN OPERATING CONTRACT

A durable workspace is a lifecycle—not a mounted directory

01 · Create	02 · Run	03 · Recover	04 · Close
seed · attach	write · checkpoint	resume · restore · fork	publish · expire · delete

Each transition needs actor • auth • idempotency • timeout • audit • cleanup

Recovery semantics

- **Commit point**
- **Write loss**
- **Remount**
- **Concurrency**

Lifecycle semantics

- **Owner**
- **Retention**
- **Publish**
- **Delete**

Create → run → recover → publish or delete

START WITH THE WORKLOAD BOUNDARY

Choose an architecture before a product

Ephemeral + publish	One-shot tasks · unexported work is lost.
Runtime checkpoint	Resume / rollback · runtime-coupled.
Durable volume	Long single-writer work · attach / zone coupling.
Embedded agent FS	Portable COW · scale and sharing limits.
Shared distributed FS	Cross-node RWX · network path and wider blast radius.

Compose, don’t rank

Image for inputs. Scratch for cache. Volume for mutable work. Artifact store for results.

The architecture is a state map—not one universal disk.

ONE STRONGEST FIT · ONE LIMITATION

The products solve different layers

Runtime-native state

Follows the runtime

AgentFS	Portable COW · beta; no shared multi-node POSIX
Sprites	Fast rollback · runtime-bound lifecycle
Modal	Managed volume · explicit commit / reload visibility

Fit: runtime semantics are part of the design.

Mounted data planes

Follows a volume

Daytona	Several layers · runtime-bound portability
Block CSI	Single writer · node / zone coupling
JuiceFS	Shared POSIX · FUSE / network / control plane

Fit: tools require a mounted filesystem.

ARCHITECTURE FIRST · ATTACHMENT SECOND

Choose the data plane, then the attachment

1 · Need shared POSIX?

Choose architecture

Tool access	Mounted POSIX required
Mobility	Resume on another node
Sharing	Shared inputs or controlled RWX

Otherwise: scratch, block, checkpoint, embedded FS, or publish.

2 · Where does it run?

Choose attachment

Direct FUSE	Sandbox owns mount + cleanup
Runtime-managed	Host owns privilege + isolation
Kubernetes	CSI / PVC; Mount Pod default; sidecar if constrained

Choose the least-complex secure lifecycle.

DEFAULTS TO TEST—NOT UNIVERSAL RULES

Start with the workload shape

Isolated work

Local or runtime-bound

One-shot	Ephemeral local + explicit publish
Long session	Runtime checkpoint or durable volume
Portable COW	Embedded agent filesystem

Shared work

Mounted data plane

Single writer	Kubernetes RWO / RWOP block
Cross-node POSIX	Shared distributed filesystem
Multiple editors	Isolate workspaces + explicit merge

Ask: What is costly to rebuild? Which boundary and RPO / RTO? Is sharing required?

If you only benchmark throughput, you have not tested durability

Kill process	Check commit point and write loss.
Kill sandbox	Restart, reattach, continue.
Lose node	Resume elsewhere; measure p95 / p99 RTO.
Kill mount	Remount; check corruption and visibility.
Partition network	Observe errors, retries, concurrent writes.

Measure what survives

Cold / warm task start

Checkpoint → restore / remount

Same-file vs. distinct-file writers

Retention, cleanup, and deletion

Verify secrets never enter snapshots or clones.

JuiceFS

RETURN TO THE FAILED SANDBOX

Preserve intent, not every byte.

Preserve what cannot be cheaply reconstructed	Isolate mutable work by default	Keep secrets and control state outside snapshots	Make checkpoint, publish, expiry, and deletion explicit	Benchmark recovery and correctness—not only throughput

Rui Su, co-founder of Juicedata

juicefs.com · rsu@juicedata.io

x.com/suavesu · linkedin.com/in/suave/

Policy first.
Mechanism second.