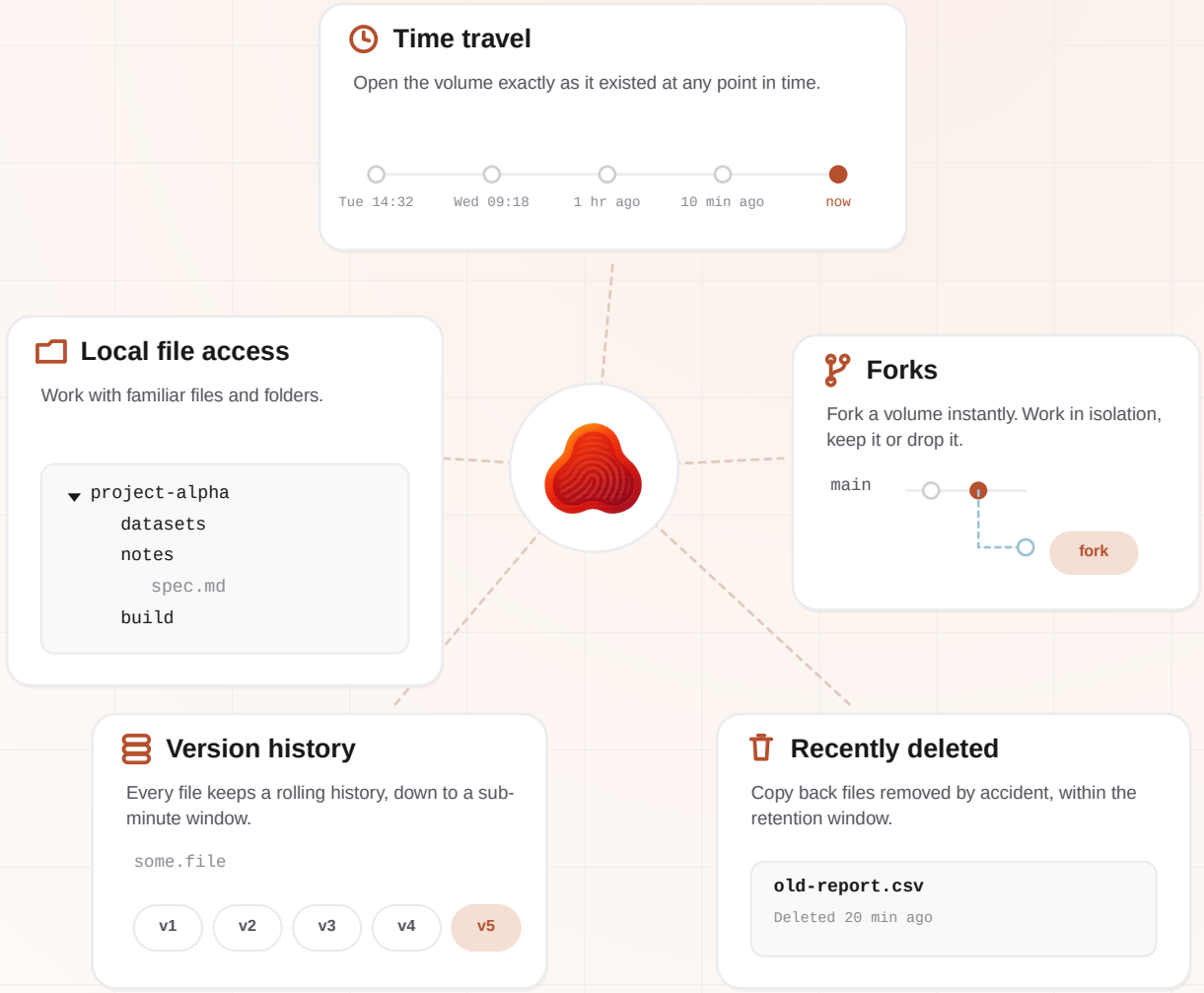
mountOS

A cross-platform, forkable, time-traveling distributed filesystem over object storage.

Mount it natively on Mac, Windows, Linux, and Kubernetes, or reach it over S3 and WebHDFS, on owned infrastructure.

```
curl -fsSL https://mountos.sh/install | bash
```



One volume, every surface.

Pick the protocol that fits the workload. They all see the same data, under the same access key.

Native mounts

FUSE, NFS, SMB, FSKit, and the mountosio driver on Mac, Windows, and Linux.

S3 API

AWS SigV4-compatible REST through the mountos gateway. Point boto3, the AWS CLI, rclone, or DuckDB at it.

WebHDFS

Local or cluster gateway, through the mountos:// Hadoop SDK.

Kubernetes CSI

ReadWriteMany PersistentVolumes, mounted into pods by the node driver.

THE SAME FILESYSTEM BEHIND EVERY ONE

POSIX semantics

Standard file and directory behavior, so existing applications and tools run as they are written.

One consistent view

Every surface reads the same volume. Completed changes are visible to clients when files are reopened.

One access key

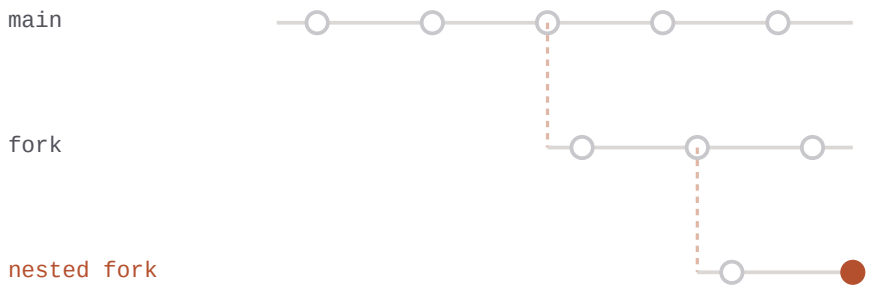
The same per-volume key authenticates a mount, the S3 endpoint, and WebHDFS.

Fork a volume, open any moment.

Branch the whole tree in one call, or open it as it looked at an earlier minute.

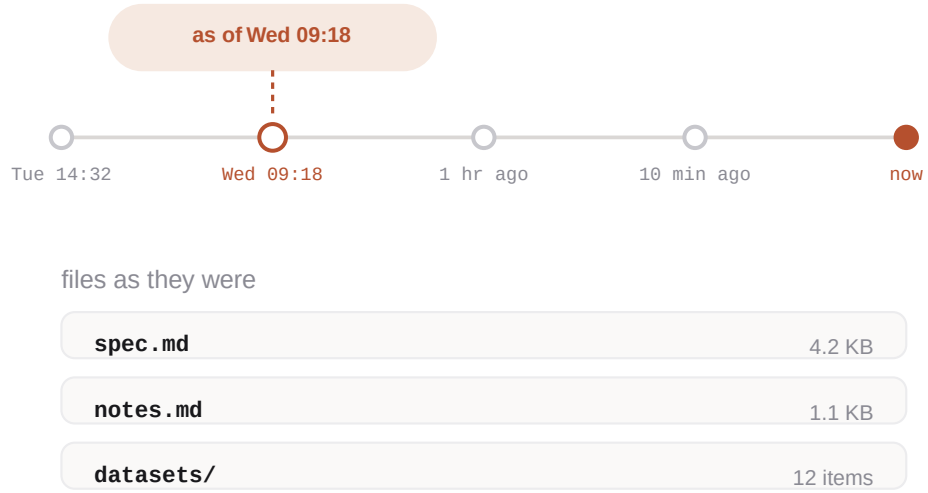
Forks

A read-write fork of an entire volume, created in one call with nothing copied. Writes stay on the fork, which can itself be forked, so branches can nest.



Time travel

A read-only, fluid snapshot of the whole volume at any point in time. Pick a timestamp down to the minute and the volume opens as it was then, ready to read or copy out.

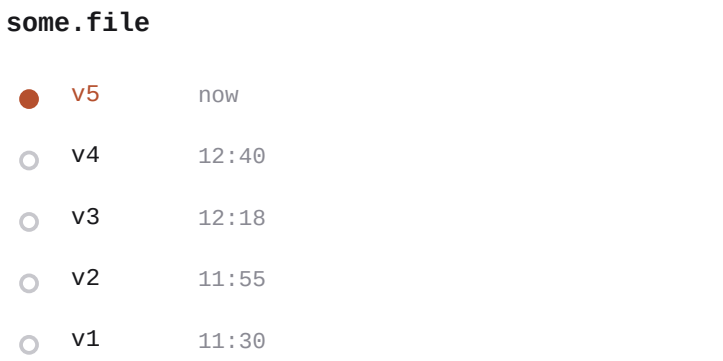


Versioned and recoverable.

Every file keeps a rolling history, and deleted files stay available for a configurable window.

Version history

Every file keeps a rolling history. One minute by default, configurable down to a sub-minute window. Saves inside the same window collapse automatically, so the history stays useful. Open any version and copy it out.



Recently deleted

Deleted files and folders stay in a recently-deleted view for a configurable window. Open any of them and copy it out. It works the same for a file, a folder, or a subtree.

Standard behavior, managed by policy.

Applications keep working as written, and one setting per volume governs the rest.

Applications run as they are

Standard POSIX file and directory behavior. Existing tools, scripts, and applications work against the volume as written.

One policy per volume

A single retention setting governs how far time travel reaches, how long deleted files stay, and when old versions age out.

Metadata has its own plane

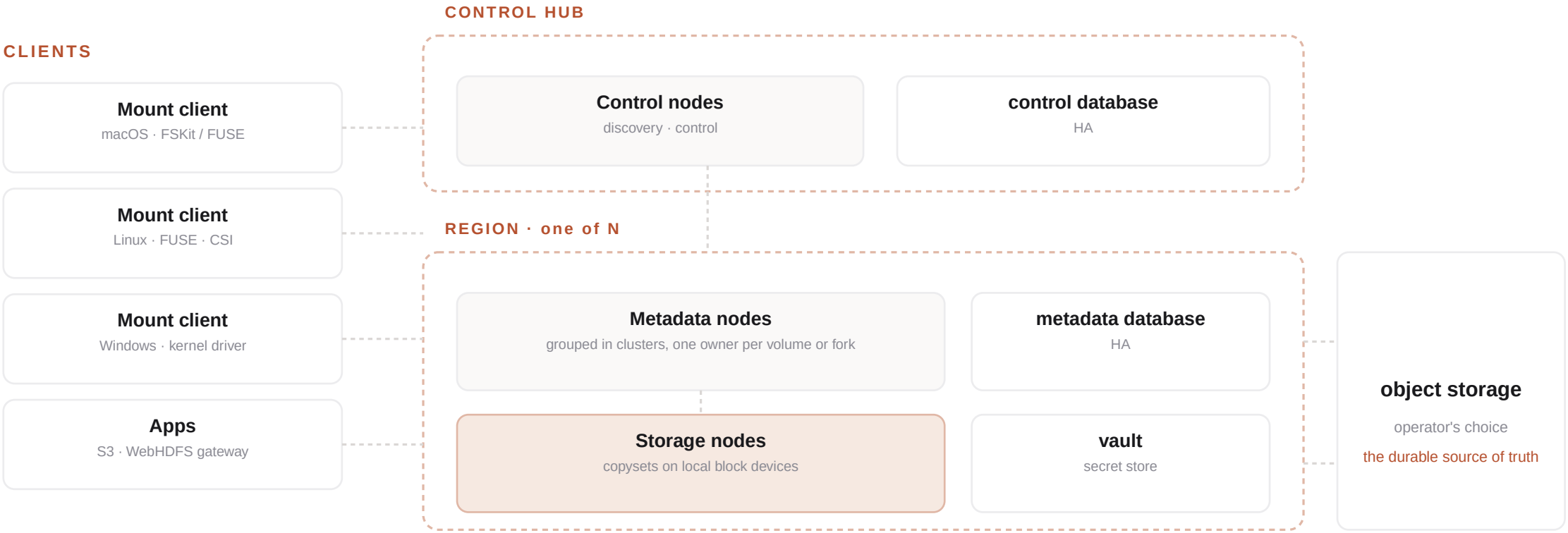
Directory operations are served from a metadata layer, held in a PostgreSQL or MySQL-wire database, so they stay fast as a volume grows.

Wire it into existing systems

An admin dashboard and an SDK in three languages, over a published REST API.

Everything runs in one footprint.

One control HUB, a region for each serving location, and owned object storage.



Block storage, for low-latency access.

A fleet of copysets in front of object storage. Writes land durably on a copyset first, then sync down.

Local-disk latency

Reads and writes land on block devices inside the region, which suits applications that need low latency.

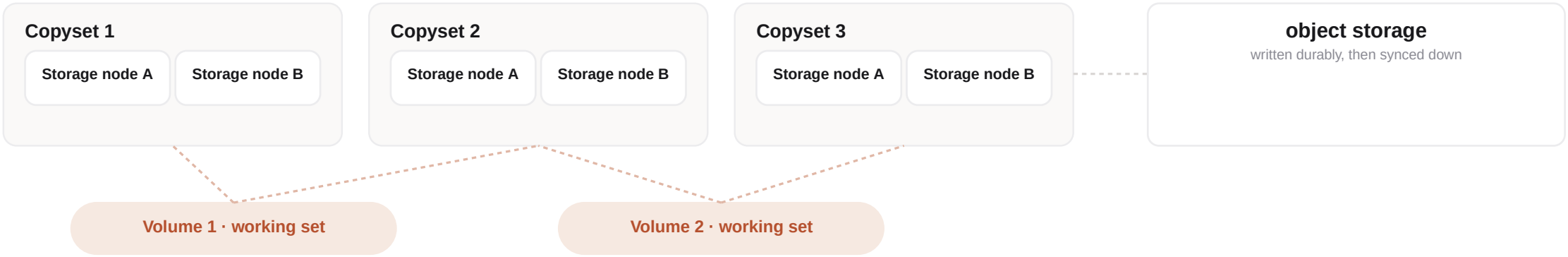
Redundant by design

Each copyset is a pair, and both members stay live and in sync.

Object storage keeps the record

Everything written settles into the object storage behind it, which stays the durable copy of the bytes.

BLOCK STORAGE IS A FLEET OF COPYSETS



Self-hosted by design.

The entire suite runs on owned infrastructure.

Where it lives

On owned infrastructure: owned servers, and the object storage already in use.

The bytes stay in place

Data stays inside one perimeter. The metadata and the content are both held there.

Recovery is built in

Version history, time travel, and recently-deleted recovery are part of the volume. Copy any version back out.

INTEGRATE IT WITH EXISTING SYSTEMS

Admin dashboard

Accounts, regions, volumes, users, and access keys, from a web interface. Open source, so it can be adapted.

Admin SDK

TypeScript, Go, and Rust packages for every administrative operation.

REST API

The same operations from any language, against a published API reference.

Made for agents and pipelines.

Connect AI agents, stream file events, and query the data as a lake.

AI MCP connector

Each client ships a read-only MCP connector, alongside a drop-in agent skill and llms.txt for any MCP-capable client.

Change-event feed

Every general volume keeps an ordered feed of creates, deletes, modifies, and renames, live or over a local REST API. Watch it to process files as they land, like generating subtitles or transcoding media.

Iceberg catalog

An Iceberg Catalog volume type presents the data as an Apache Iceberg catalog, so analytical engines query it directly.

SANDBOX MOUNTS

Mount a temporary, writable fork of a volume, pinned to the live volume or to any point in its history. Experiment freely, then unmount and every change is discarded.

Verified like a real filesystem.

POSIX compatibility is checked with the industry filesystem suites, each covering a different surface.

8,789

pjdfstest cases, for POSIX compliance

Standard file and directory semantics let existing applications and tools run as they are written.

✓ **LTP**	filesystem and system-call behavior
✓ **xfstests**	filesystem regressions
✓ **fsx**	I/O consistency under randomized operations
✓ **IOR + mdtest**	I/O throughput and metadata load

A clear behavior contract.

What mountOS guarantees, governed by the configured retention window.

Durable

Committed data remains safely stored through failures and restarts.

Available

Storage continues serving through some individual node failures.

Elastic

Capacity grows naturally with the amount of data stored, without requiring fixed volume sizing.

Consistent

Completed changes are visible to clients when files are reopened.

Recoverable

Earlier states and deleted data remain accessible within the retention period.

Isolated and regional

Independent workloads remain separated, while data and storage operations stay within their region.

Get started

The entire suite is self-hosted. One installer pulls the binaries. Provision a HUB, a region, and a volume, then mount.

```
$ curl -fsSL https://mountos.sh/install | bash
```


john@mountos.io · support@mountos.io