NeuralMesh Replication
Technical Brief

NeuralMesh Replication

NeuralMesh Replication breaks the dependency between where data lives and where AI workloads run. It separates namespace visibility from data movement, giving remote clusters immediate access to the source filesystem while moving only the data each workload requires. The result is a single architecture for workload mobility, multi-site collaboration, data protection, and migration, without full dataset copies or object storage as an intermediary.

Executive Overview

GPU capacity is scarce, expensive, and constantly shifting across regions and providers. Teams reserve capacity in advance, seize spot opportunities, burst into the cloud, and move workloads wherever the economics and availability are best. But securing the compute is only half the challenge. Training datasets reach petabyte scale. Checkpoint directories grow with model complexity. The data behind AI workloads cannot travel as freely as the compute that needs it. The result: GPU capacity opens in a new region or cloud environment, but the data required to run the job is anchored elsewhere. Teams stage, copy, or manually move data before work can begin.

Traditional replication was designed for a different problem: creating full secondary copies for disaster recovery and continuity. That model holds up when storage is centralized and compute is predictable. At AI scale, across dynamic GPU environments, it creates friction at every step. Destinations are not useful until copies complete. Storage costs multiply with site count. Teams build custom staging pipelines to handle movement that replication should cover natively.

NeuralMesh Replication breaks that dependency. This brief describes the architecture: how clusters pair and establish trust, how metadata and file data move independently, how hydration policies give teams control over what becomes local and when, and how a single replication foundation supports workload mobility, multi-site collaboration, data protection, and migration.

Replication Model

NeuralMesh Replication is snapshot-based and asynchronous. The source cluster remains authoritative throughout an active replication relationship. Replication advances by shipping snapshots on a configurable cadence; each snapshot captures the filesystem state at a point in time. The destination cluster maintains a read-only view of the source filesystem while the relationship is active.

The model separates three things traditional replication ties together: when the destination becomes useful, when file contents arrive, and how often the destination gets an updated view. Source metadata (directory structures, file names, sizes, permissions, and file pointers) syncs to the destination first, making the namespace browsable before any file data has been transferred. Metadata update cadence and hydration policy are then configured independently. This sequence changes how teams interact with remote data. Instead of waiting for a full copy before work can start, teams browse the source namespace immediately, identify what a workload needs, and hydrate exactly that.

	Traditional Replication	NeuralMesh Replication
Destination visibility	Destination becomes useful after the full copy completes.	Metadata syncs first; the filesystem is browsable before file contents arrive.
Data locality	Every destination is treated as a full copy.	Choose the right data-locality model: cache on demand, hydrate selected paths, or maintain a full copy.
Policy control	Copy schedule determines both data freshness and data movement.	Metadata update cadence and hydration policy are configured independently.
Data path	Object storage or staging pipelines may serve as handoff layers.	Metadata and file contents move directly between WEKA clusters.

If a replication relationship is paused or disrupted and later re-established, synchronization coordinates from a common snapshot. A full retransfer is not required. RPO is bounded by the snapshot interval. Teams size the interval based on recovery objectives and available WAN bandwidth.

Cluster Pairing and Trust

Before replication runs, two clusters establish a trust relationship over a secure management channel. Trust is established bidirectionally. TLS is mandatory for all communication; authentication uses certificates. Once established, the trust relationship is visible to cluster administrators on both sides and covers all subsequent Filesystem Pair (FSP) operations between those clusters.

Trust establishment requires administrator privileges on both clusters. FSP creation requires filesystem administrator rights on both source and destination. An audit log covers all replication operations, providing an accountable record for security and compliance governance.

The control and data planes stay cleanly separated. The management channel handles trust establishment, FSP lifecycle, and status reporting. Filesystem metadata and file data never traverse it. Both use S3 protocol over HTTPS, which keeps the control plane lightweight and avoids coupling replication data movement to the management channel.

FSP Relationships

Once clusters trust each other, teams create FSP relationships that map a source filesystem to a destination filesystem. Each FSP is directional: source to destination. Bidirectional and cascading replication (source to destination to a third cluster) are not supported in the current release, which keeps the consistency model straightforward and prevents failure amplification across chains.

FSP relationships move through defined lifecycle states: created, active, paused, and terminated. Replication pauses automatically during cluster upgrades, then resumes after a version compatibility check. Teams can pause and resume relationships manually for maintenance, migration, or bandwidth management.

One source filesystem can replicate to multiple destination clusters, supporting fan-out patterns for multi-site deployments. The destination filesystem is read-only while replication is active. If a destination filesystem is written to directly, the replication relationship breaks and the system alerts.

Namespace-First Visibility

When replication begins, source metadata syncs to the destination before file contents arrive. The destination cluster receives directory structures, file names, sizes, permissions, and file pointers, enough to present a POSIX-accessible view of the source namespace. Teams browse the dataset at the destination, understand what exists, and plan data movement before any file data has been transferred.

Metadata sync is atomic per snapshot. A snapshot's metadata either completes at the destination or it does not. There is no partial state that presents an inconsistent view. Metadata is pushed from the source cluster's S3 Gateway to the destination cluster's S3 Gateway over HTTPS.

Why namespace visibility changes the operating model

Traditional replication makes data movement a prerequisite for planning. NeuralMesh Replication

inverts that sequence.

- Destination namespace is browsable immediately after metadata sync, before any file data arrives
- Teams identify what a workload needs before committing WAN bandwidth or destination storage
- On-demand hydration starts from a real access pattern, not a preemptive full copy
- Multi-site teams plan and prioritize independently, without blocking on each other's transfers

Hydration Engine

File contents arrive at the destination based on the hydration policy configured for each filesystem. Three policies address different operational needs, and all three run on the same replication foundation. Teams do not need separate infrastructure for workload mobility, collaboration, and recovery.

On-Demand Hydration

On-demand hydration retrieves and caches files when a workload first accesses them at the destination. The file is not present locally until requested; once fetched, it is cached for subsequent access. This policy suits workloads where access patterns are unpredictable or where only a subset of a large dataset will be read. The workload does not wait for files it will never touch.

Selective Hydration

Selective hydration proactively copies specified files or directories to the destination before a workload begins. Teams configure which paths to hydrate. This works well when the access pattern for an upcoming job is known: a pre-training dataset directory, specific model checkpoints, or defined inference inputs can be staged in advance so the workload reads data locally from the first access. Progress is tracked per file; operations are resumable if the relationship is disrupted and re-established.

Full Hydration

Full hydration maintains a complete local copy of the source filesystem at the destination. This is appropriate for recovery scenarios, migration to a new primary site, or operational requirements where the entire dataset needs to be locally available.

Policy	When to use	Destination storage required
On-demand	Unknown or partial access patterns; GPU cloudbursting	Only accessed files, minimal until workload runs
Selective	Known access pattern; latency-sensitive or pre-staged jobs	Selected paths only
Full	Recovery; migration; complete operational mirror	Full dataset capacity, equal to source filesystem

Data Flow

The replication architecture separates control, metadata, and data across distinct channels. Control operations (trust establishment, FSP lifecycle, status reporting) run over a dedicated management

channel with TLS 1.2 or higher. Filesystem metadata and file data both use S3 protocol over HTTPS, moving in opposite directions and on separate timelines.

Source metadata is pushed: the source cluster's S3 Gateway pushes snapshot metadata to the destination cluster's S3 Gateway. Each cluster runs an internal S3 Gateway component. No external object storage is required or used as an intermediary. This is a direct cluster-to-cluster transfer. Object storage can participate in tiering and other data services but is not in the replication path.

File data is pulled: the destination cluster's S3 Gateway fetches file contents from the source cluster's S3 Gateway. Pull-based transfer aligns with the hydration model; the destination determines when and which data to fetch, driven by policy and workload access. Data integrity is verified via checksums. Incremental sync between snapshots transfers only changed blocks, not the full dataset on each cycle.

S3 Gateway Component

Each WEKA cluster running replication includes an internal S3 Gateway that handles the transport layer. On the source, the S3 Gateway exposes snapshot metadata and file data for the destination to access. On the destination, the S3 Gateway imports metadata pushed from the source and serves data pull requests to the Hydration Engine.

S3 was chosen for both metadata and data transport because it decouples replication traffic from the management channel's data structures, tolerates version skew between clusters running different WEKA versions, and uses the same transfer mechanism proven in WEKA's snap-to-object workflows. Replication between clusters at different patch versions does not require tight API alignment.

Bandwidth and Resource Management

Replication bandwidth is configurable at two levels. Per-FSP limits cap throughput for an individual replication relationship. A global cluster-wide cap protects production workloads from aggregate replication traffic across all active relationships. Both controls can be adjusted at runtime without disrupting active replication.

Separate throttling controls apply to metadata push and data pull operations. A burst of metadata push activity at snapshot time has a different profile than sustained background data hydration; teams may want different ceilings for each.

Snapshot retention on the source cluster is coordinated with the destination. The source retains replicated snapshots until the destination confirms successful data sync. This prevents data loss if the relationship is disrupted before hydration completes for a given snapshot. Source snapshot retention policy should be sized to accommodate the expected sync completion window.

Security

All control plane communication uses TLS 1.2 or higher. All metadata and data transfer uses HTTPS. Certificate-based mutual authentication applies to both JRPC and S3 channels. S3 access authentication is configurable per cluster pair.

The separation of cluster administrator and filesystem administrator roles applies to replication management. Cluster administrators manage trust relationships and infrastructure. Filesystem administrators manage FSP relationships within their scope. Neither role requires the other's credentials to operate.

The destination filesystem is read-only while replication is active. Direct writes to the destination break the replication relationship and trigger an alert on both sides. This enforces source authority and prevents split-brain scenarios.

Monitoring and Observability

NeuralMesh Replication exposes telemetry covering replication lag and RPO tracking, bandwidth utilization across metadata push and data pull, hydration progress and queue depth, snapshot sync success and failure rates, S3 Gateway operation latencies, and data pull throughput.

Events and alerts cover trust relationship status changes, FSP state transitions, replication lag exceeding a configured threshold, version compatibility warnings, S3 Gateway connectivity failures, metadata push failures, data pull failures or timeouts, and data integrity mismatches.

NeuralMesh Observe integration pushes replication telemetry for centralized monitoring, supports cross-cluster replication topology visualization, and retains historical RPO and throughput metrics across replication relationships.

Failure Handling and Resilience

NeuralMesh Replication handles network disruptions without requiring full retransfer. Metadata push failures trigger source-side retries with exponential backoff. Data pull failures trigger destination-side retries. Both operations queue requests during S3 Gateway unavailability and resume when connectivity is restored.

Snapshot consistency is maintained atomically per snapshot. Destination snapshots are marked incomplete until metadata sync finishes. There is no window where a partially-synced snapshot appears complete. Hydration status is tracked per file; crash recovery resumes from the last checkpoint rather than restarting.