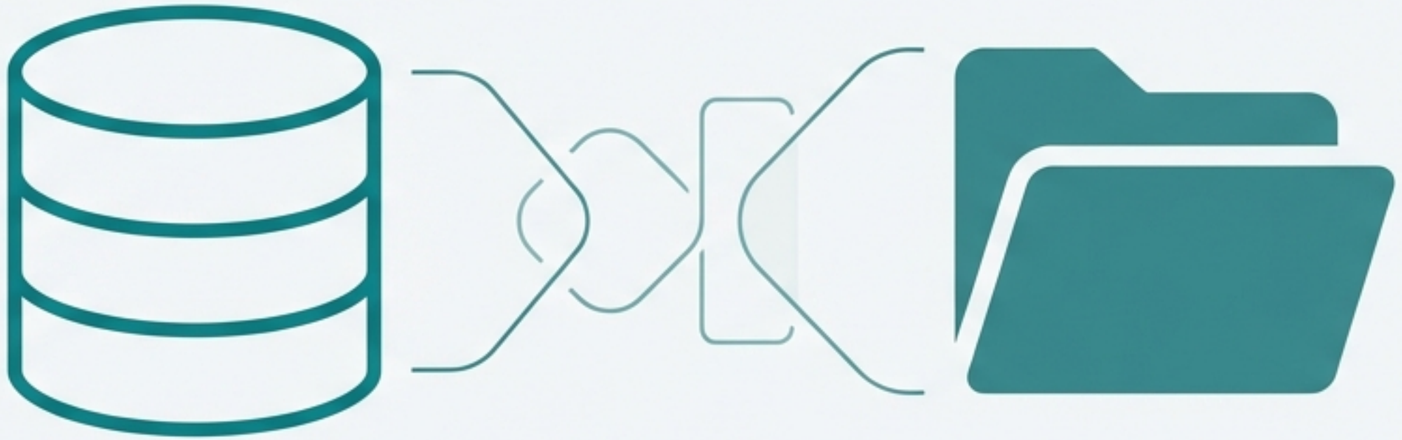
Your Next Database is a File System.

How modern cloud storage offers a simpler, more scalable, and cost-effective foundation for data-intensive applications.

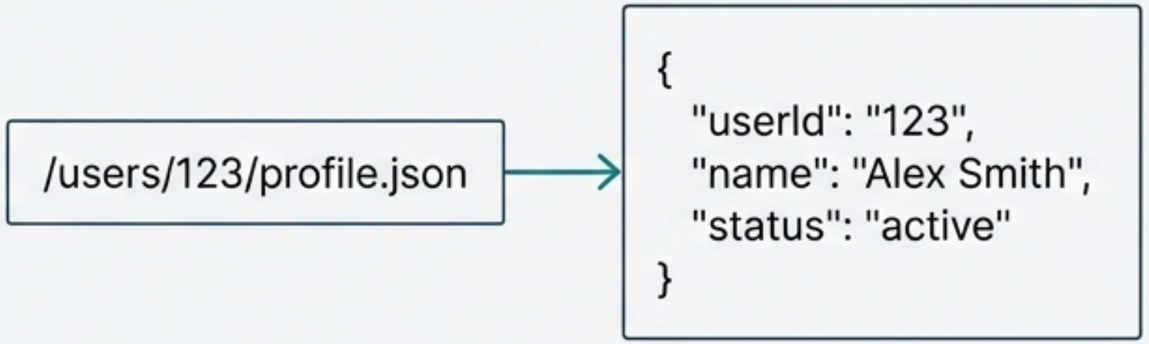


> This deck argues for a fundamental shift: treating cloud object storage not just as a data lake, but as a powerful, serverless, key-value database for a wide range of modern workloads.

The Foundation: A File System is Already a Key-Value Store.

At its core, a file system is a hierarchical key-value store. Modern cloud storage (S3, Azure Blob) extends this with a flat, highly scalable namespace.

- **Key:** The file path
- **Value:** The file's content



> Amazon S3 is a very large key–value store: the key is the filename, the value is the contents of the file.
>
> *– John Rotenstein, Stack Overflow*

For any application that needs to store a value and retrieve it by a unique key, cloud storage is a natural and effective database out of the box.

The Operational Shift: From High-Maintenance Pets to Scalalable Cattle.

Traditional Database

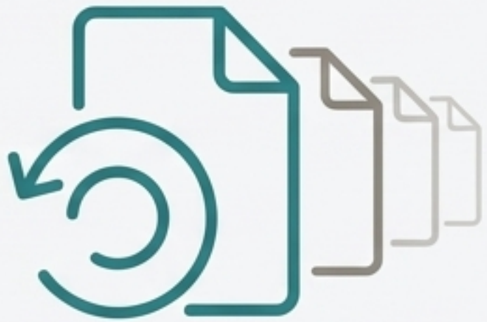
- Requires constant care, patching, and monitoring.
- Scaling is complex and expensive.
- A single point of failure that can fail at 2 AM.

Cloud Storage as a Database

- Managed at scale by the cloud provider.
- Stateless from your perspective; no server to fail.
- Pay nothing when idle; always available on-demand.

Using cloud storage as your database removes the burden of running a dedicated database server, letting you focus on your application, not on infrastructure.

Resilience and Durability are Built-in, Not Bolted On.



Automated Replication & Durability

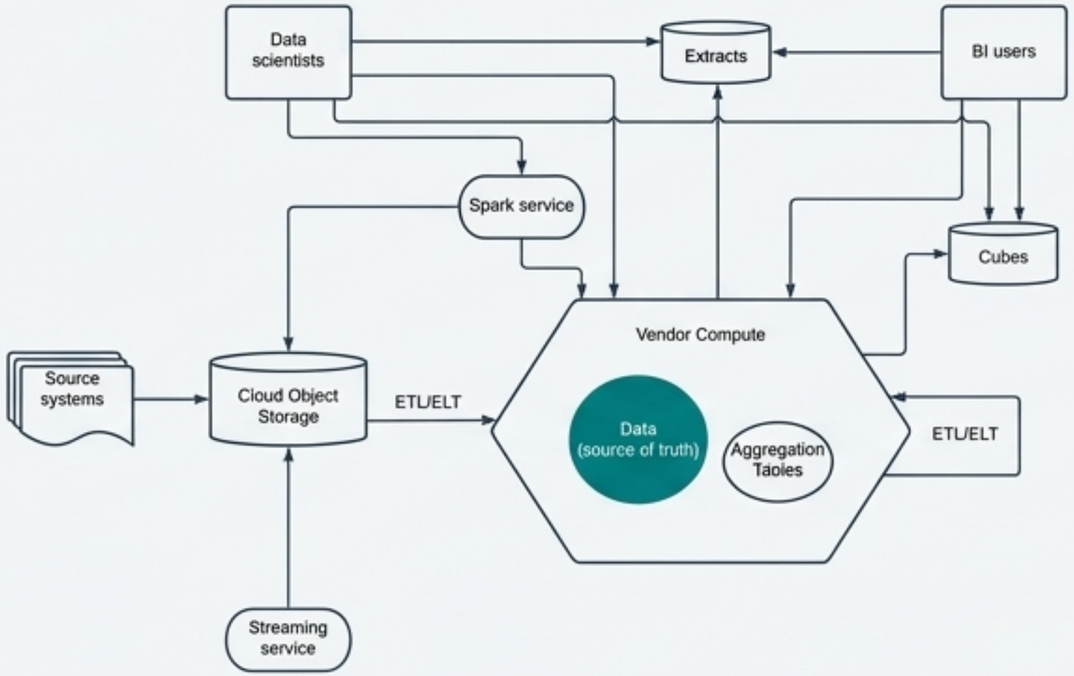
- Cloud object storage inherently replicates data across multiple servers and availability zones.
- This provides ‘enhanced durability’ without any configuration or management overhead.
- Your data is protected from infrastructure failure by default.

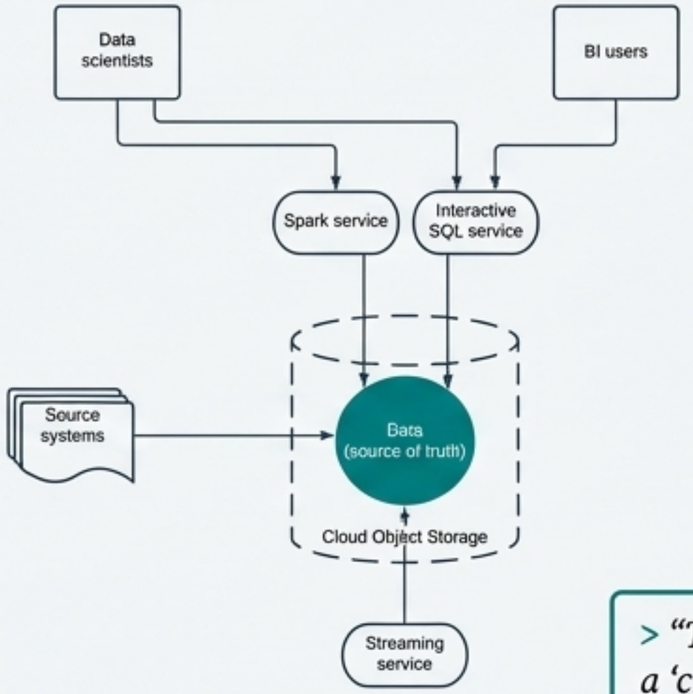
Effortless Versioning & Backup

- Features like point-in-time versioning can be enabled with a single click.
- Older versions of objects are retained automatically, protecting against accidental deletions or overwrites.
- Eliminates the need for complex backup jobs and failover server management.

The administrative overhead of ensuring data safety and availability is effectively offloaded to the cloud platform.

The Architectural Revolution: Decoupling Compute from Storage.





> *"This separation was a 'critical step forward for efficiency' in cloud architecture."*

FROM → **TO**

Coupled Architecture. Data and processing are locked inside a monolithic database or data warehouse. To scale one, you must scale the other.

Decoupled Architecture. Data lives in a central, scalable cloud storage layer. Multiple, ephemeral compute services (SQL, Spark, ML) access it on demand.

Benefit 1: Unlocking Near-Infinite Scale and Radical Cost Efficiency

Massive, Linear Scalability

- Object storage is designed for “unlimited growth to petabytes” and can handle billions of keys.
- Scaling is simple: just add more objects. No sharding, re-indexing, or provisioning bigger hardware.
- Pay-for-what-you-use model means you never pay for unused capacity.

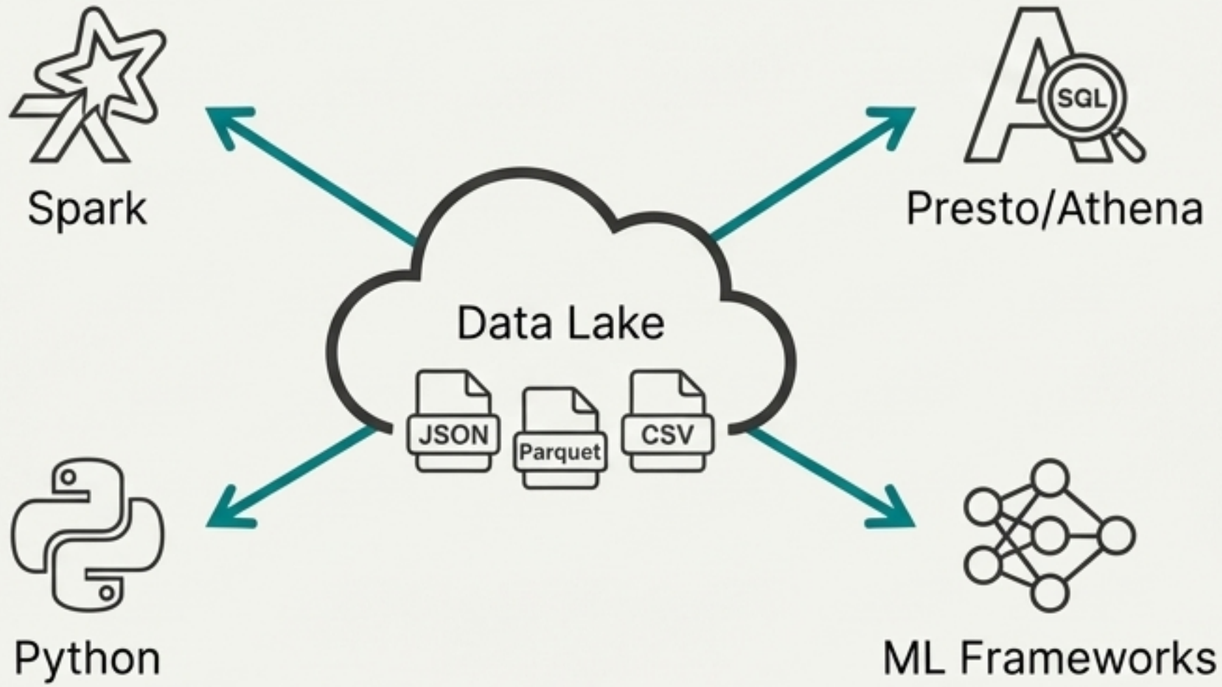
Dramatically Lower Costs

- Raw storage costs per gigabyte are a fraction of high-performance database storage.
- Quote: “raw storage costs became so cheap they were practically ‘free’ relative to other IT costs.”
- Isolating compute means you only pay for processing when you actually run a query.

Cost Comparison: While slower for some lookups than a managed NoSQL DB like DynamoDB, Amazon S3 costs significantly less for storage, making it ideal for large datasets.

Benefit 2: Gaining Flexibility and Freedom from Vendor Lock-in

When data lives in open formats (Parquet, JSON, CSV) on a universal storage layer, you are not tied to a single vendor's query engine.



Key Advantages:

- **Bring Any Tool to the Data:** Use a SQL engine for BI, a Spark cluster for ML, and a custom script for ad-hoc analysis—all on the same source data.
- **Future-Proof Your Architecture:** Easily adopt new, faster processing frameworks as they emerge without complex data migrations.
- **Universal Access:** Any language or platform with an SDK or REST client can interact with the data.

Decoupling "allows for universal data access from unlimited services and applications."

The On-Demand Pattern: Compute in Bursts, Not as a Service

1. LOAD

A request arrives. Ephemeral compute (e.g., a serverless function) spins up and loads necessary data from cloud storage into memory.

2. PROCESS

Complex queries, graph traversals, or transformations run at high speed on the in-memory data.

3. UNLOAD

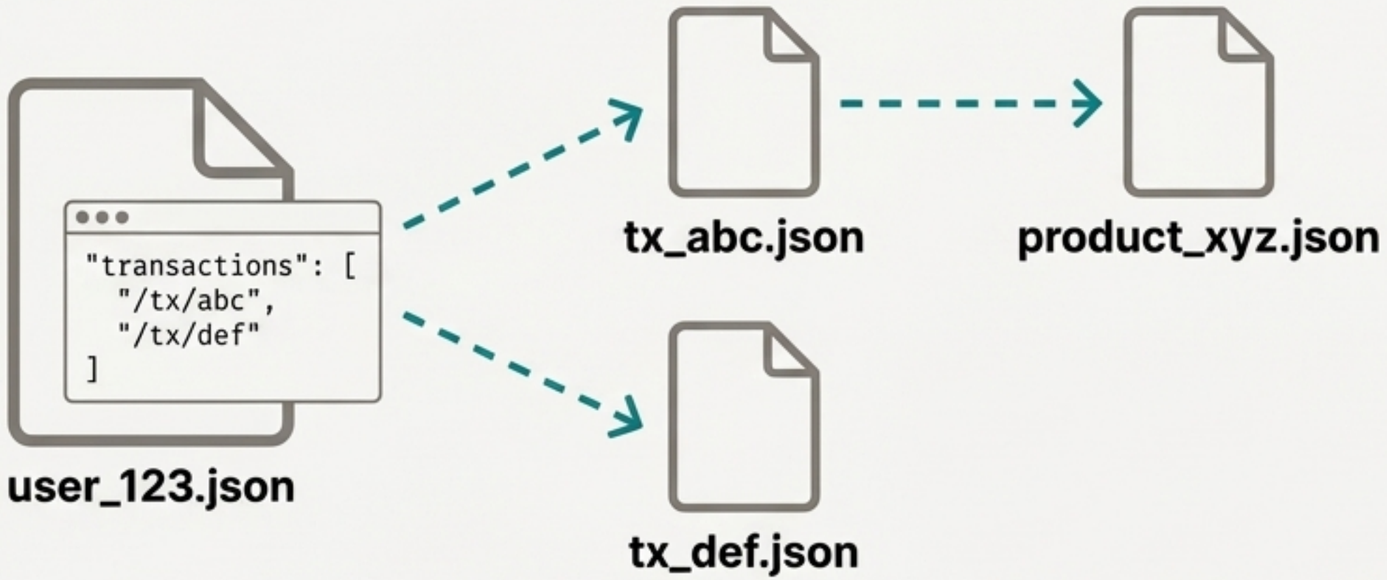
Results are returned, and the compute instance and in-memory data are discarded. The authoritative state remains in storage.

Key Characteristics

- **Truly Serverless:** “Nothing is running until a query or operation is initiated.” You only pay for active compute.
- **Stateless Workers:** Compute instances are cattle. They can be destroyed and recreated at will because the data lives externally. This simplifies deployments and scaling.
- **Fast Enough:** Reading hundreds of MB or a few GB from cloud storage into memory is often a matter of seconds, making this pattern viable for many interactive workloads.

Beyond Key-Value: Building Complex Graphs on a Foundation of Files.

Relationships are not defined by a rigid schema, but by creating “hyperlinks” between files. A file’s content can contain references (paths or IDs) to other files.



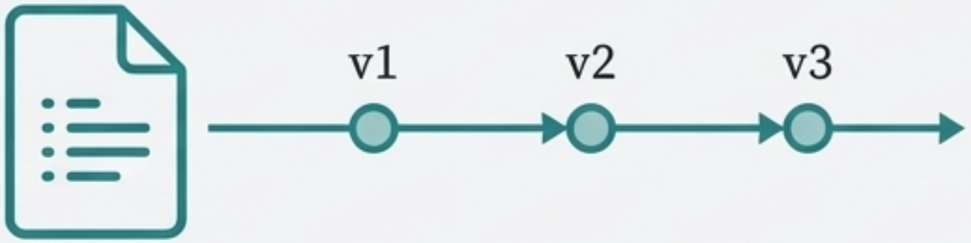
How it Works

- **Nodes:** Each file represents a node in the graph (e.g., a user profile, a product).
- **Edges:** References inside the file content act as edges, creating a network of interconnected data.
- **Traversal:** Querying relationships becomes a matter of reading a file and following the links to read subsequent files.

This approach encourages a `schema-on-read` model where the structure adapts to your data, rather than forcing data to conform to a database schema.

The Unmatched Power of Files: Immutable Versioning and Provenance

Fine-Grained Version Control



- Each file can be treated as an immutable record.
- Cloud storage's built-in object versioning creates an automatic audit trail of every change.
- You can link a file to its previous version, creating a complete historical chain.

Transparent Provenance for AI

- Data is not hidden in a monolithic binary blob. You can inspect any file at any time.
- This is invaluable for explainability. An AI system can justify an output by pointing directly to the source file(s) it used.
- Quote: "it's straightforward to keep an audit trail of changes."

****AI Agent Context****
Files serve as a transparent "memory," where new facts are appended as new files, creating a clear chain of reasoning.

This Isn’t a Silver Bullet: Acknowledging the Trade-offs.

Cloud storage is eventually consistent and lacks native support for complex, multi-object transactional guarantees. It’s crucial to know when a traditional database is still the right tool.

Use a Traditional Database When You Need:

Strict ACID Transactions: Operations that must succeed or fail together across multiple records (e.g., financial ledger).

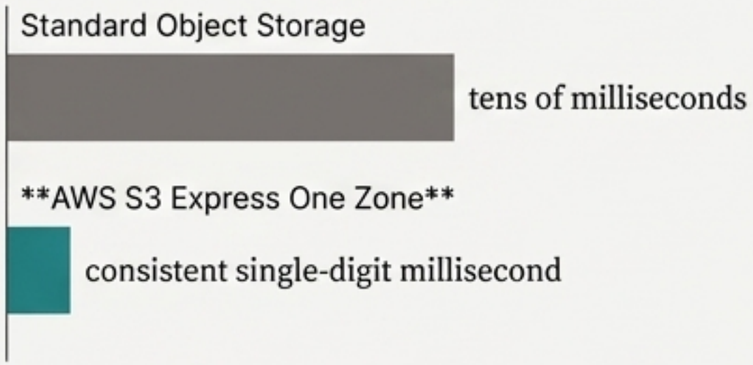
Real-time Complex Queries: High-performance JOINs and aggregations across millions of records with sub-second latency.

High-Frequency Small Updates: Workloads with thousands of small (e.g., <4KB) read/write operations per second.

> *“When you do need transactions or a query DSL, then you don’t really have a choice.”*

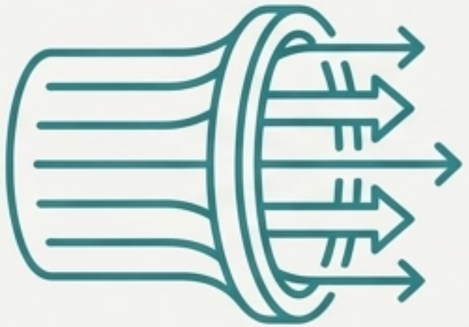
The Performance Conversation: Latency, Throughput, and Modern Realities

Latency



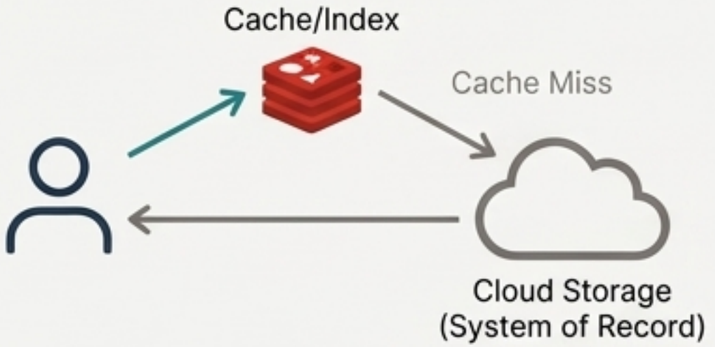
- **The Trade-off:** Standard object storage has higher first-byte latency than a local database.
- **The Innovation:** New high-performance storage classes are purpose-built to deliver low latency, closing the gap significantly.

Throughput



Throughput for large data streaming can be extremely high, as reads of many objects can be easily parallelised.

Hybrid Solutions



For the best of both worlds, combine cloud storage as the system of record with a front-end cache or index to accelerate lookups for hot data.

A Simpler, More Scalable Default.

Simplicity & Resilience

Offloads maintenance, replication, and versioning to the platform.

Scale & Economics

Offers limitless, pay-as-you-go storage at a fraction of the cost of database servers.

Flexibility & Freedom

Enables decoupled architecture, use of best-of-breed tools, and avoids vendor lock-in.

The goal isn't to replace every database. It's to challenge the default. By starting with the simplest, most scalable persistence layer, you build systems that are more resilient, cost-effective, and adaptable to future needs.

Unless there are data relations to exploit, a relational database is just a tool looking for a problem.