LucidLink

TECHNICAL BRIEF

LucidLink Filespaces
Technical Overview

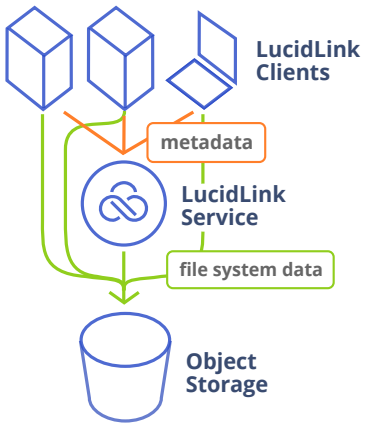
LucidLink Filespaces provide a distributed, log-structured file system layer for object storage, allowing simple and high performance access.

The main issues with storage today are that it is siloed, expensive, and ages out quickly.

Trying to realize the promise of the cloud, organizations are changing the way that they access and consume data. However, reality gets in the way. It is hard to centralize and manage storage. It is hard to refactor disparate systems to consume one unified dataset; one single source of truth. And it is even more challenging to control infrastructure cost.

We combined concepts from distributed databases, log-structured file systems and cloud gateways to completely rethink cloud storage; building a cloud-native file system on top of object storage.

LucidLink believes that by creating the best of both worlds, we address a highly sought-after solution for a setup where users and applications can connect to the same single namespace, exposed as local disk but backed by object storage. Highly scalable and inexpensive object storage can be consumed as a local file system layer that allows users and applications to access this from anywhere. This can be used to replace on-premises file servers, allow distributed teams access to their data, and provides ways to connect applications, cloud regions and cloud providers all to the same dataset.

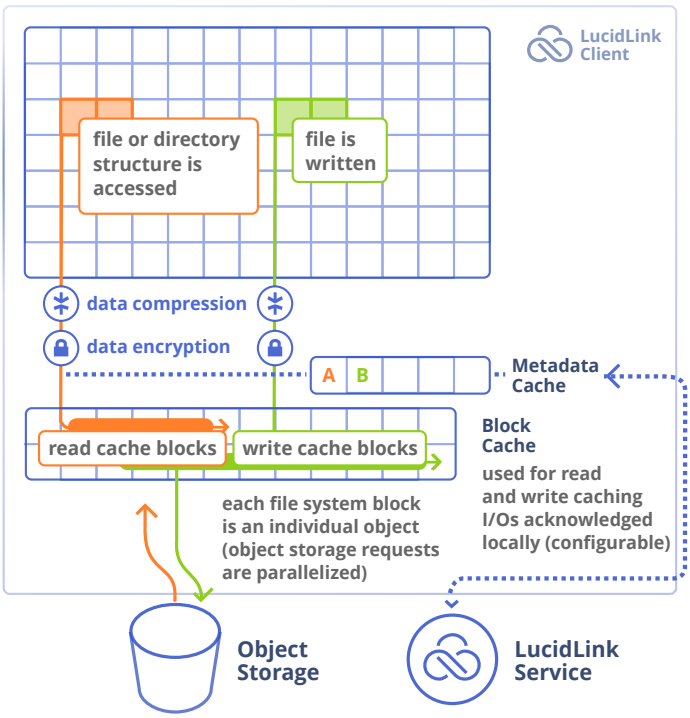


The Filesystem

LucidLink Filespaces involve a hub and spoke topology architecture, with a virtually limitless set of clients connected to object storage and a LucidLink Service. All you need to do is provision an object storage bucket on your favorite cloud service, and we add the magic that transforms it into scale-out, and share-able volume.

To connect to the Filespace a LucidLink Client talks to both the LucidLink Service holding the metadata key value store, and to the object store holding the file system data. The LucidLink Client creates a local file system mount point.

Separating metadata and file system data allows the Filespace to persist across all connected clients. Files are split into individual blocks, where the metadata provides the index allowing instant access, no matter where your data is located.



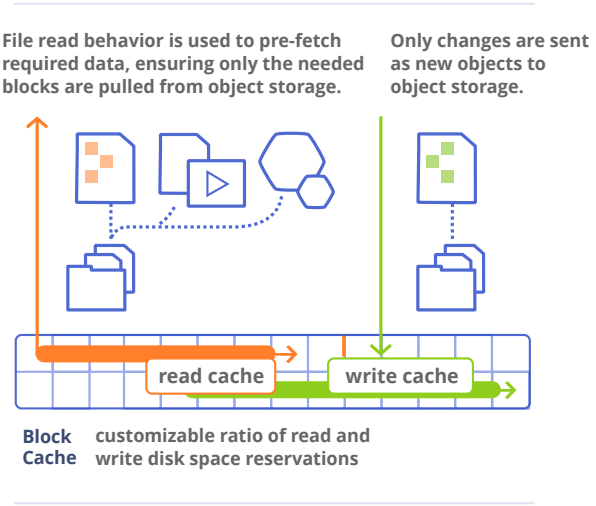
Read and Write Caching

LucidLink Filespaces come with a configurable block size, used to determine file block boundaries.

While accessing file blocks sequentially or when traversing a directory structure the LucidLink Client pre-fetches blocks to meet demand. This allows you to stream only the parts of a file you need. When editing a dataset, this allows you to work on only the part of the data that is required without wasting bandwidth, local disk space, and most importantly, eliminating the time spent waiting for the entire dataset to synchronize.

Once a file is written on the client, metadata operations have immediate effect, and as the file uploads to object storage, metadata is updated further to reflect changes. As the file system data can take some time to transmit to the object storage a local cache is used. Writes are acknowledged locally giving a consistent file system experience.

Read and write cache ratios and reservations can be configured, allowing this to be optimized to your specific application or workload.

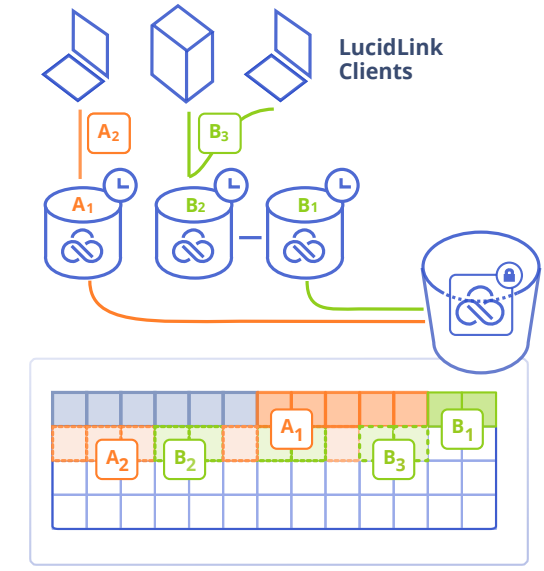


Snapshot Support

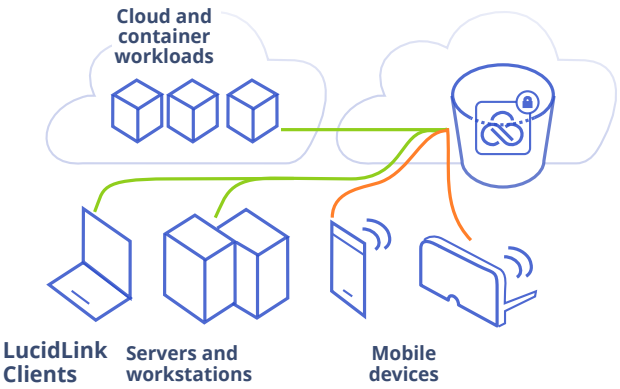
Snapshot support comes natural to LucidLink as Filespaces are built on a log-structured file system. The file system metadata keeps track of individual client edits and file versions. A new block is written per edit allowing complete version control.

Snapshots are one of the biggest differentiating features in the industry, leading to the growth of the entire software and storage virtualization verticals. LucidLink takes this one step further allowing cloud-native snapshot support for object storage buckets, without requiring expensive block storage or complicated tiering.

Simply snap and clone your cloud-native Filespaces with no performance impact. Mount and access a different version of your Filespace for dev and test, restore or compliance reasons. When a snapshot is no longer required blocks are garbage collected out.



Object storage is a stored in a distributed fashion so the fragmentation of block data that is typically caused by snapshots is not an issue. Instead multiple spindles and disks across a scale-out object storage solution are used to keep linear performance as the Filespace grows.



To create the mount point a FUSE driver is used on Linux-like systems, and an equivalent driver is used on Windows.

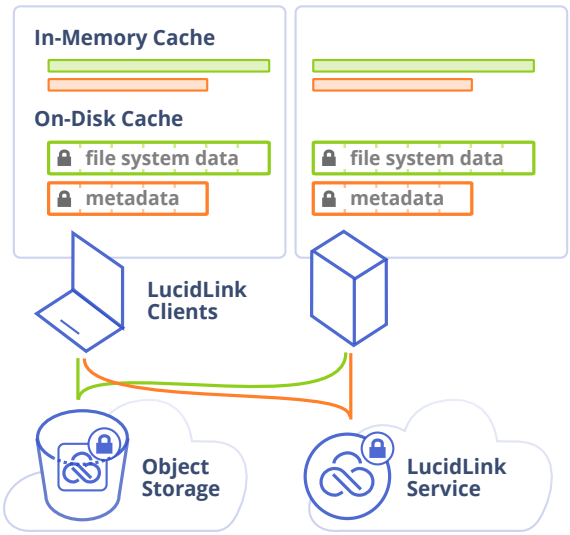
Multi-OS Support

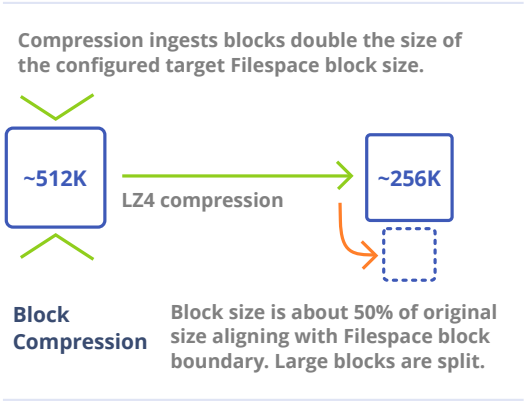
The LucidLink client is available for Windows, MacOS, Linux as well as Android and iOS. It is perfectly suited for anything from hosting large analytics datasets, cloud container workloads, traditional application servers, to use cases such as front-ending network share user data, IoT device log gathering and streaming content for virtual reality applications.

Data Encryption

LucidLink Filespaces use AES-256 encryption. Every object stored is encrypted, metadata included. This means that neither LucidLink nor the object storage provider has visibility of data in the Filespace.

LucidLink Client cached metadata and cached file system data is encrypted at rest. This helps resolve challenges with distributed teams. Instead of downloading files on individual laptops, or storing specific data sets for application use outside of a cloud sharing or network file service, data remains securely in the Filespace. Depending on the configuration, the data actively being used is cached locally, and if the LucidLink Client disconnects, so does the ability of the client to access locally cached data.





Data Compression

When writing a block to a LucidLink Filespace LZ4 compression is used. Depending on the block size and data type this results in a compression ratio of 2-2.5:1. This means a 50% reduction in disk space for uncompressed and unencrypted data. Already compressed data such as video, audio may not benefit from being compressed further. Encrypted files cannot be compressed.

After LucidLink Filespace blocks are compressed they are encrypted and are put in the write cache and written to object storage.

Initializing a Filespace

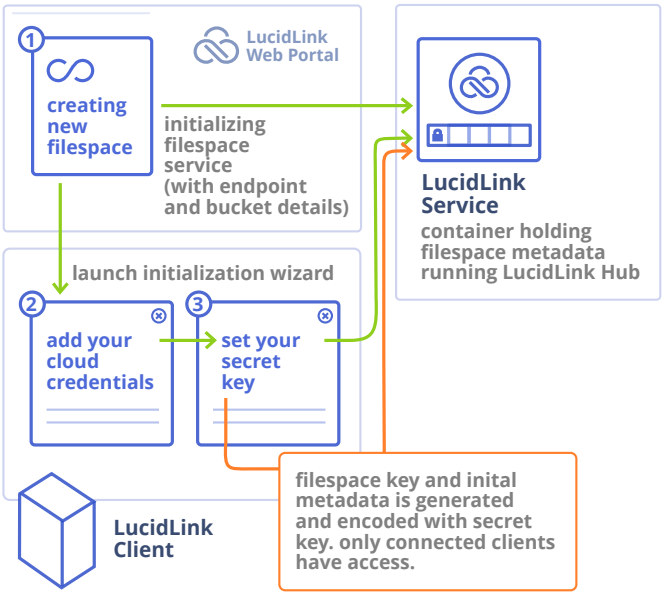
When creating a Filespace in the LucidLink web portal a new instance of the LucidLink Hub Service is started. This container-based service is what is used to host the Filespace metadata key value store.

After that the LucidLink Client is downloaded and used to initialize the Filespace. The client requests cloud credentials to be used by both the client and LucidLink Service to access the file space object data.

Next the client requests a shared secret. This key is only available on the LucidLink Client side and would be used by other clients to access the same Filespace.

The initialization process is split between the LucidLink Web Portal and the local LucidLink Client to ensure LucidLink never has access to the secret key. This secret key is not stored anywhere except in memory when the LucidLink Client is connected. If this secret key is lost there is no way to recover the Filespace data.

LucidLink Client initialization can also take place via the command line, allowing custom parameters such as custom block sizes. In rare cases it is also possible to run the LucidLink Service on-premises.



How-to Initialize a Filespace via the Command-line

```
lucid daemon

lucid init-s3 --fs <filespace.domain> --password
<sharedsecret> --endpoint <ipaddress/url:port>
--access-key <accesskey> --secret-key  <secretkey>
--https --region <region> --block-size <KiB>
--bucket-name <bucketname> --provider <vendor>
```

Connected Clients

LucidLink Filespaces are designed to allow multiple clients to connect to the same Filespace. In large scale environments this easily outperforms traditional file servers as each client machine has its own cache, metadata and connection to the object store. This is useful for use cases with distributed workflows; large teams of users, or a group of distributed applications all connected to the same data set.

For some workloads, such as legacy file server replacement or use as a backup repository, it may make sense to have only a few clients connected to the same Filespace. These servers running the LucidLink Client may be used to share-out S3 data using traditional network protocols, such as CIFS and NFS. It makes sense to significantly increase the LucidLink client cache size for this scenario.

When connecting to a LucidLink Filespace metadata is synchronized. This allows fast file access and applications to seek within files for specific blocks only.

As files are acessed objects are prefetched from object storage. If the access pattern to either a file byte range or directory is sequential, this triggers the prefetcher. LucidLink Clients each have individual connections to object storage allowing for efficient access.

When writing to a LucidLink Filespace each file write is versioned with a client identifier. These changes are cached locally before being uploaded. Metadata is always synchronized first before file system data is written to the object storage.

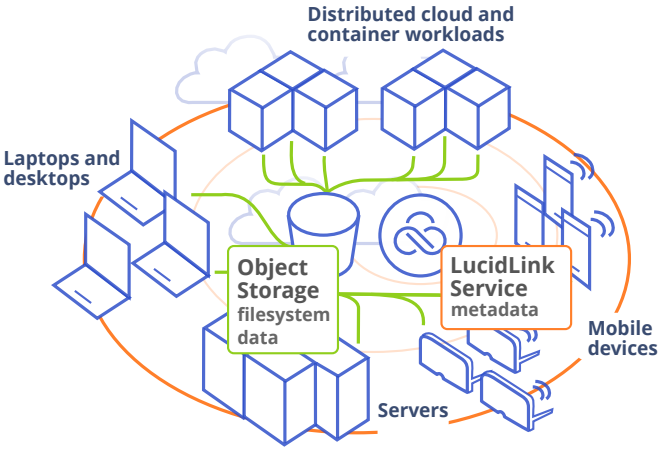
Metadata is shared with the LucidLink Service which centrally arbitrates Filespace access. Actual file system data writes happen directly from the LucidLink Client to the object storage. Because LucidLink is a log-structured file system any new writes result in the creation of a new object.

Garbage Collection

As LucidLink is log-structured and keeps track of block and file versions it is possible to go back in time using snapshot support.

Each new write turns into a new object on object storage. LucidLink never updates existing objects, ensuring this does not interfere with object storage versioning and that LucidLink always has access to the latest copy of data.

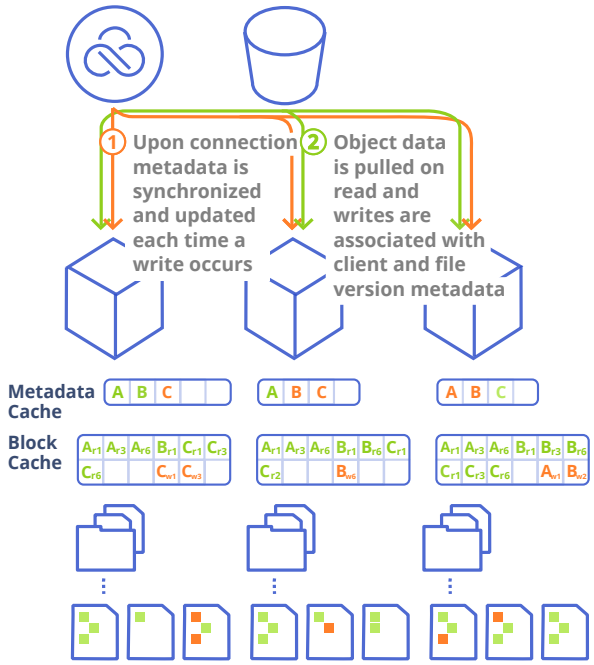
Stale objects are garbage collected. The LucidLink Service runs garbage collection every 10 minutes by default.



LucidLink Clients

Virtually limitless scale-out is enabled by splitting out file system data on object storage from metadata in the LucidLink Service.

Improved performance comes from the ability to pre-cache client reads, track and tie client writes to individual clients and send this directly to highly-scalable object storage.



Read data is prefetched or pulled straight from object storage.

Writes are acknowledged locally before being transmitted to the object storage.

LucidLink Service arbitrates write access. Write related metadata has priority over file system block data and is transmitted first.

Creating The Service You Need

The LucidLink Filespace architecture is flexible. Each component in the architecture is modular, allowing individual Filespaces to be created with or without compression or encryption, with different I/O acknowledgement settings, different block sizes, different cache ratios or custom connection limits, and more.

Over time as new features are added to LucidLink Filespaces these follow the same extensible architecture, allowing for a highly scalable file system with exactly the functionality you need.

LucidLink aims to build tools and products that address customer need when it comes to cloud storage adoption. Good products are never built in a vacuum, and at LucidLink we value feedback from our customers. Customer support is a big part of that, but more importantly it is our goal to listen to your input and create products that include your suggestions.

We believe cloud object storage has the power to fundamentally change the way individuals and businesses store and access their data. This stretches from simple data sharing to consumption of object storage by distributed applications. We are at the forefront of object-based cloud storage and services.