

# Storage, Zwischenlösung LfULG



Version 1.1 (2025-08-25)

## Vorbemerkung

Der Storage-Server von Artefactual kann über sogenannte Spaces eine oder mehrere Locations definieren. Jede Location definiert ein zugeordnetes Dateisystem. Die Workflow-Pipelines von Archivemata werden auf die Locations des Storage-Servers gemappt.

Ein transparenter Umzug von Daten von einem auf einen anderen Space ist nicht vorgesehen und erfordert einen AIP-Reingest.

Locations können als 'replica' definiert werden. In dem Fall wird bei Speichern auf eine Location, die 'replica'-Location ebenfalls angesprochen. Der Storage-Service meldet nur dann einen Erfolg, wenn die Kopie auch auf die Replica-Location geschrieben werden konnte. Eine nachträgliche Definition von 'replica'-Locations ist möglich, gilt aber nicht für AIPs, die schon gesichert wurden. Um diese zu replizieren, muss ebenfalls ein AIP-Reingest angestoßen werden.

Aus der bisherigen Erfahrung ist es sinnvoll, die Migration von Storage mitzudenken. Das Storage-Modell, welches der Storage-Server von Artefactual definiert, ist nicht hinreichend, da AIP-Reingests eine hohe Processingbelastung für Archivemata bzw. Storage-Server generieren.

Sinnvoller ist es, mindestens einen S3-Server davorzusetzen. Dies erlaubt eine feingranulare und flexible Konfiguration. Die korrekte Zuordnung von AIPs erfolgt durch die notwendige Festlegung von Buckets, die persistent in den Spaces des Storage-Servers hinterlegt sind.

## MinIO - Möglichkeiten und Grenzen

Die Software MinIO <sup>[1]</sup> ist eine quelloffene <sup>[2]</sup> Implementierung eines Objektspeichersystems mit S3-Kompatibilität. Das Projekt hat eine aktive Community und wird durch Player, wie Intel, Dell, Nvidia, VMWare, Accenture und anderen unterstützt.

In MinIO werden jedem S3 Server initial entweder Verzeichnisse oder andere MinIO Server <sup>[3]</sup> als Volumes zugeordnet. Jedes Volume enthält die kompletten Informationen <sup>[4]</sup> um, beim Verlust von anderen Volumes, diese vollständig wiederherzustellen.

Volumes müssen initial beim ersten Start eines MinIO Servers festgelegt werden. Eine **nachträgliche** Änderung ist **nicht** möglich. Sollen Volumes wachsen können, so wäre Folgendes machbar:



1. ein S3 Knoten wird abgeschaltet
2. dessen lokale Volumes werden durch welche mit größerem Speicherplatz ersetzt
3. der S3 Knoten wird wieder eingeschaltet, die Replikation auf das neue Volume wird durchgeführt

4. Wiederhole ab Schritt 1 bis alle S3 Knoten erweitert

Alternativ kann LVM für die lokalen Volumes genutzt werden.

Buckets sind Namensräume, die ein logisches Laufwerk bilden. Schreib- und Leseinstellungen für Objekte werden pro Bucket definiert.



Da Buckets logische Datenträger darstellen (Namensraum) kann später auf die Daten auch bei einer anderen zugrundeliegenden Architektur zugegriffen werden.

Dabei wird durch MinIO sichergestellt, dass jedes Objekt, dem Quorum <sup>[5]</sup> entsprechend, in den Volumes gespeichert bleibt. Objekte werden dabei anhand eines Keys identifiziert und in den Volumes unter dem Bucket-Namen und dem Key gesichert.



#### Quorum

Das Schreibquorum beträgt bei MinIO  $q=(N/2)+1$ , das Lesequorum  $q=N/2$ , jeweils bezogen auf  $N$  - Anzahl der Volumes.

Volumes dürfen unterschiedlicher Größe sein, solange für jedes Objekt sichergestellt ist, dass es so in den Volumes gespeichert werden kann, dass das Quorum abbildbar ist.

#### Beispiel 1. Quorum und Objektgröße

Wenn von drei, einem Bucket zugeordneten, Volumes noch zwei 1GB freien Speicher haben, aber eines schon voll ist, so kann in den Bucket noch ein Objekt der Größe ~1GB geschrieben werden.

Das langsamste Volume, welches für die Erfüllung des Quorums benötigt wird, bestimmt hierbei die Schreibgeschwindigkeit.

MinIO kann keine Konsistenz garantieren, wenn für Volumes NFS verwendet wird <sup>[6]</sup>.



MinIO's strict read-after-write and list-after-write consistency model requires local drive filesystems.

MinIO cannot provide consistency guarantees if the underlying storage volumes are NFS or a similar network-attached storage volume.

## Vorläufiges Stagesystem LfULG

Um die bisherige Konfiguration des ZIH vorläufig weiter nutzen zu können, wird, wie in Abbildung [Systemskizze Minimal](#) zu sehen, ein S3 Server vorgeschaltet.

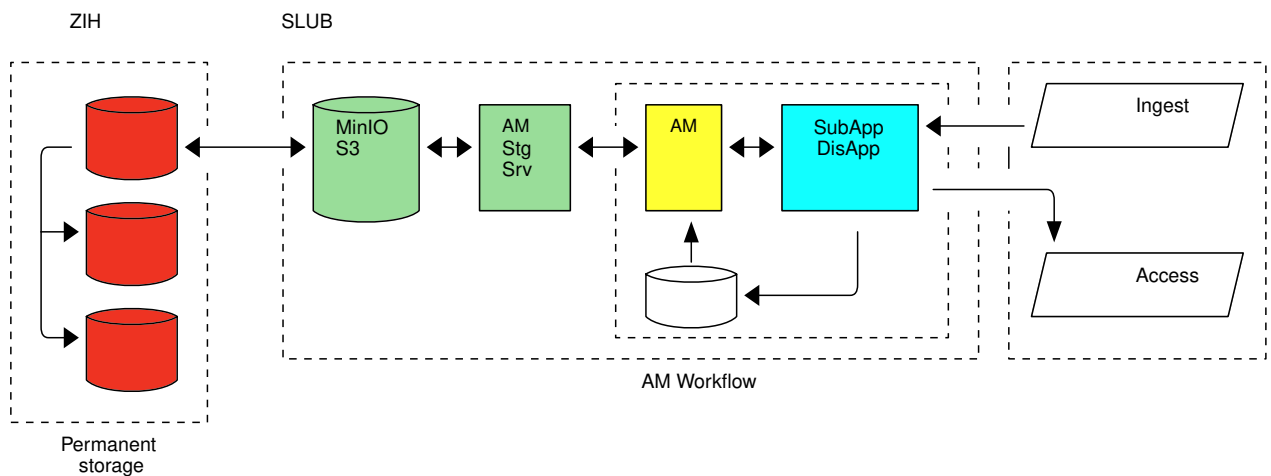


Abbildung 1. Systemskizze Minimal

Der Zugriff von Archivematica auf den **/permanent** Storage erfolgt via Archivematica Storage Service. Dort ist jeder Pipeline eine Storage Location zugeordnet, im vorliegenden Fall, wie in Abbildung **Minimalvariante Minio S3** sichtbar, eine, die einen S3 Space nutzt. Der S3 Space wird durch eine Storage ID identifiziert. Der Storage Service verwendet für diesen S3 Space das S3 Protokoll, um über das Bucket **LZA** auf die im MinIO S3 zugeordneten Volumes zuzugreifen.

Diese Volumes werden durch NFS basierte Mountpoints realisiert, die vom NFS Server des ZIH den Zugriff auf den GPFS Speicher des ZIH erlauben. Die Replikation erfolgt in dem Fall erst auf ZIH Seite.

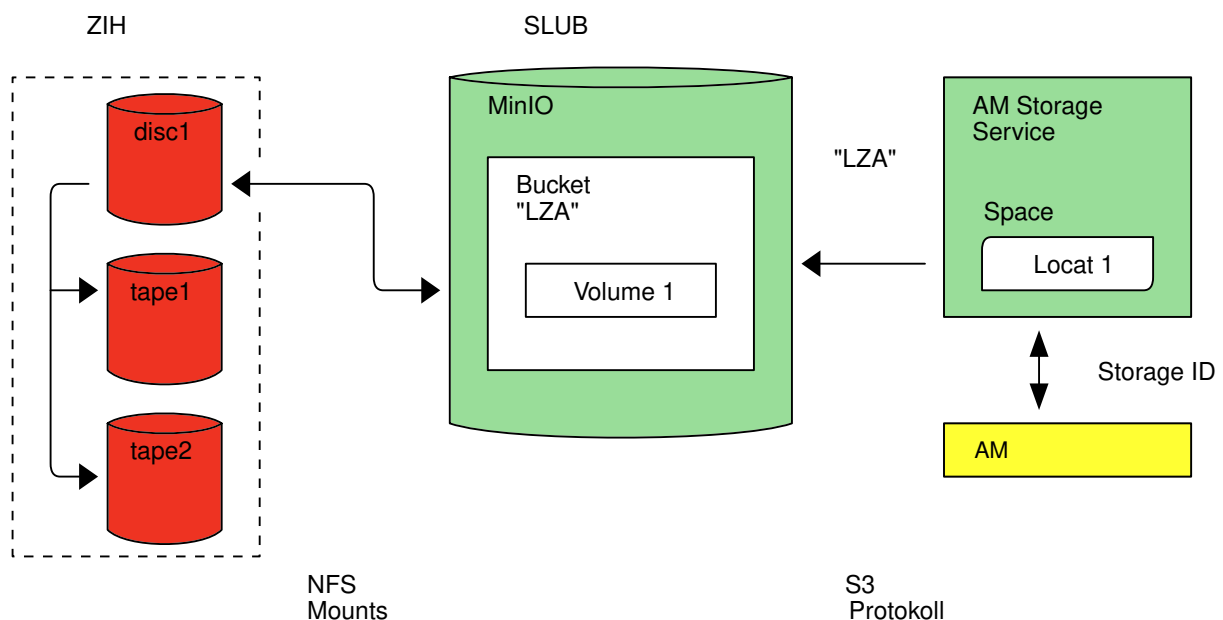


Abbildung 2. Minimalvariante Minio S3



Die Variante aus **Minimalvariante Minio S3** ist **nicht** zu empfehlen, da

- die Verwendung von NFS basierten Volumes dem Konsistenzmodell von MinIO nicht genügt

- der Migrationsaufwand höher ist!

In Vorbereitung auf eine spätere Migration würde man den einzelnen nfs-Mountpoint aus [Minimalvariante Minio S3](#) benutzen, **darin** Unterverzeichnisse anlegen und jedes der Unterverzeichnisse einem Volume eines S3 Servers zuordnen.

## Migration Storage System LfULG

Besser wäre es, die Mounts vom ZIH **einzeln herauszulösen** und den Volumes zuzuordnen, wie in Abbildung [Variante 2 mit separaten Mounts vom ZIH, Minio S3](#) zu sehen.

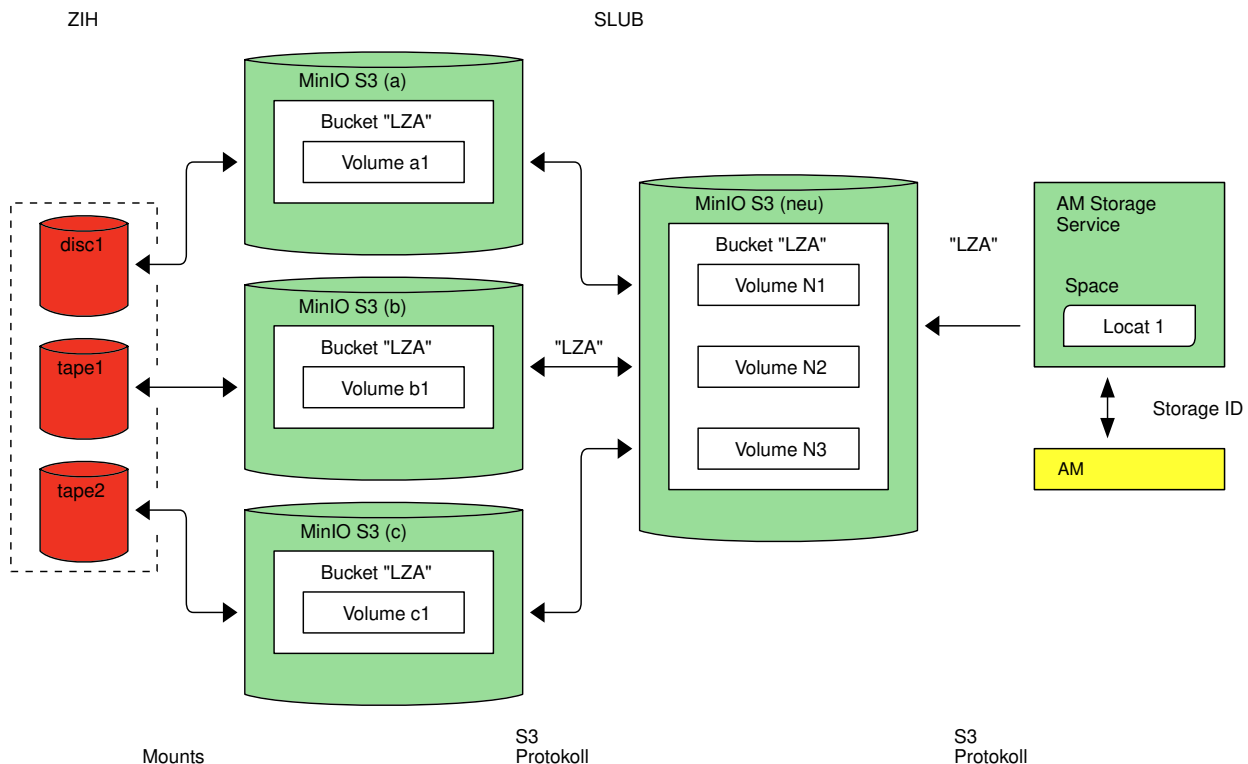


Abbildung 3. Variante 2 mit separaten Mounts vom ZIH, Minio S3

Die Migrationsschritte wären dann folgende:

1. Sicherstellen, dass jedes Kopienlaufwerk auf ZIH Seite durch uns mountbar ist
2. Anlegen von drei S3-Servern (a, b, c) mit je mindestens einem Volumen, welches auf Laufwerk im ZIH verweist
3. Anlegen eines S3 Servers ('neu') mit drei Volumes, welche auf a, b und c zeigen
4. Bulk-Kopieren von altem S3 'alt' auf neuem S3 'neu' via `mcli mirror alt/lza/ neu/lza`
5. Ändern der S3-Adresse im Storageserver von 'alt' auf 'neu'



Besser als diese Migration wäre es, von vornherein die Variante in Abbildung [Variante 2 mit separaten Mounts vom ZIH, Minio S3](#) einzusetzen

# Zielvariante, flexible Nutzung von beliebigem Storage via S3 Protokoll

Die wünschenswerte Variante wäre dann die, dass im Storage Service mehrere Locations hinterlegt werden, die jeweils via S3 angebunden sind und den Bucket (hier: *LZA*) teilen.

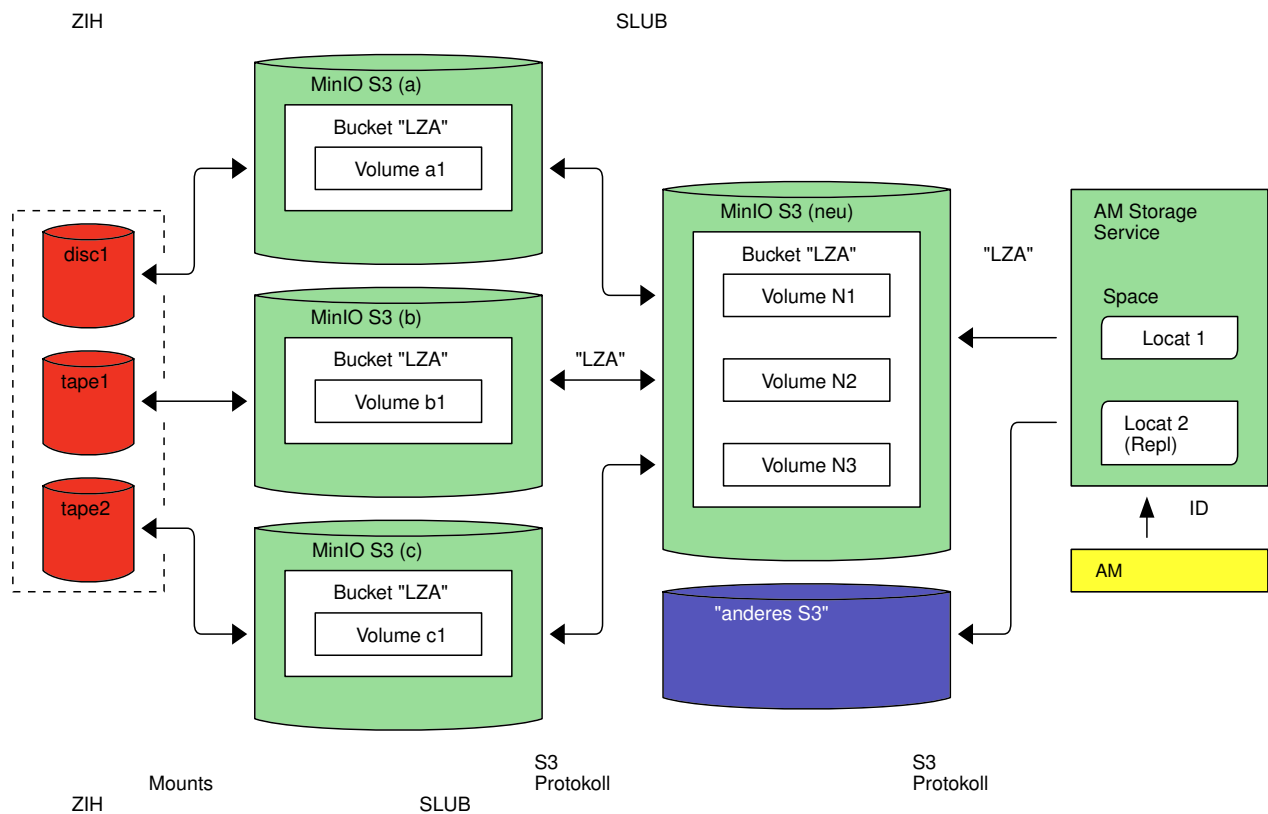


Abbildung 4. Variante 3, erweitert minimal, Minio S3

Die Variante aus Abbildung [Variante 2 mit separaten Mounts vom ZIH, Minio S3](#) lässt sich leicht in Zielvariante nach Abbildung [Variante 3, erweitert minimal, Minio S3](#) überführen, da dann der MinIO Server einfach nur ein S3 Provider unter mehreren ist. Die Location *Locat1* und der Bucket bleiben hierbei gleich (der *space* muss aber angepasst werden)

Die S3 Provider können ihren Speicher beliebig organisieren, solange das zugewiesene Bucket (hier: *LZA*) verwendet wird.

Das Modell [Variante 3, erweitert minimal, Minio S3](#) entspricht dann im Wesentlichen der Abbildung [Systemskizze Endzustand 1](#).

Die einfachste und günstigste Implementierung wäre, wenn die S3 Provider für ihre Volumes das Linear Tapefilesystem (LTFS) nutzen würden. In dem Fall müsste den Locations ein Replica Storage zugeordnet werden. Ähnliches Verhalten lässt sich erreichen, wenn auf S3 Server Seite, passende Volumes für Tape- und Diskspeicher hinterlegt werden, wobei der Diskspeicher analog zum Replica Storage funktioniert.

Soll auch ein unterschiedliches Protokoll für den Zugriff auf Storage verwendet werden, so könnte

man S3 Provider und NFS Provider (oder andere Protokolle) in den Locations passend hinterlegen. In dem Fall ist die Austauschkompatibilität durch fehlenden, gemeinsamen Namensraum <sup>[7]</sup> nicht mehr gegeben.

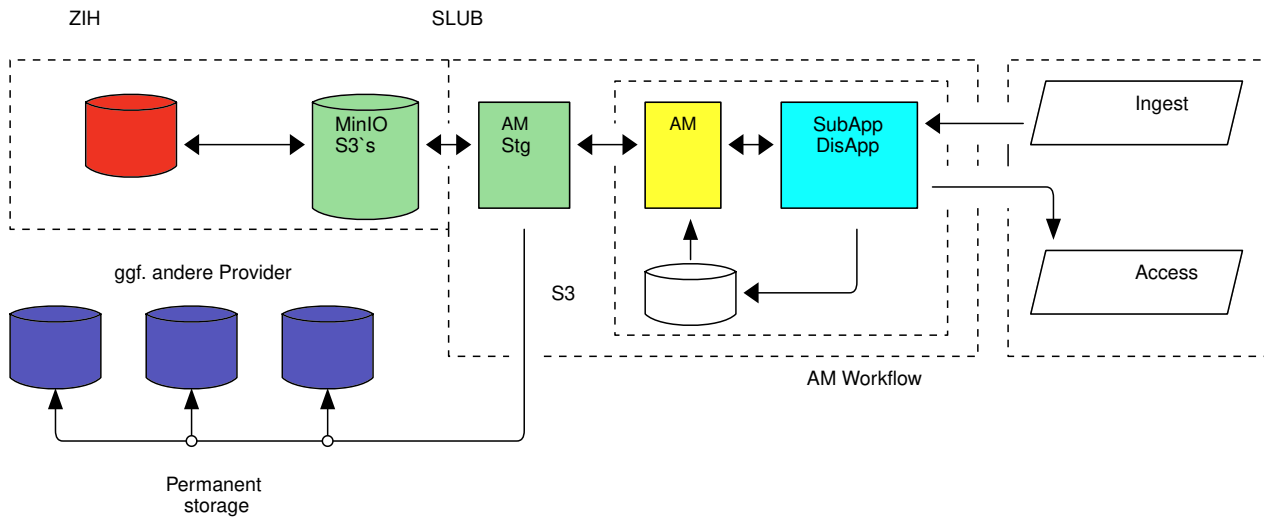


Abbildung 5. Systemskizze Endzustand 1

Alternativ wäre die Variante [Systemskizze Endzustand 2](#) denkbar. Dies hätte den Vorteil der leichteren Replikation <sup>[8]</sup>. Dabei werden die sonstigen Provider nicht über die Locations des Storage servers, sondern über einen einzelnen MinIO <sup>[9]</sup> verwaltet.

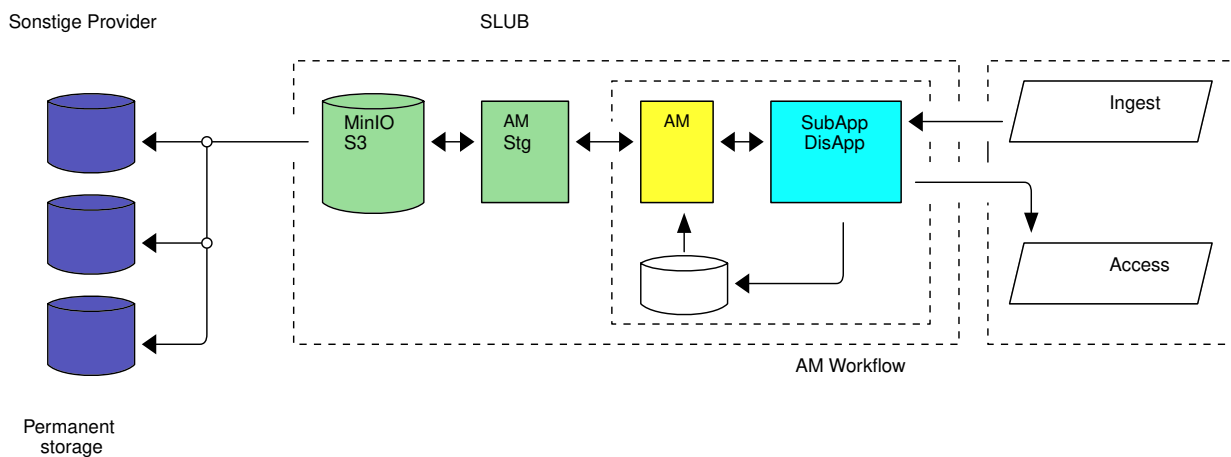


Abbildung 6. Systemskizze Endzustand 2

In der Skizze [Systemskizze Endzustand 2](#) nicht ersichtlich, könnte MinIO und Storage server so konfiguriert werden, dass Redundanz erzeugt wird, indem zwei Locations definiert werden, die je auf einen eigenen MinIO zeigen <sup>[10]</sup> und/oder die MinIO im Cluster-Betrieb konfiguriert werden.

# Redundanz

In der digitalen Langzeitarchivierung sollten folgende Prinzipien zur Bitstreamsicherung von Objekten eingehalten werden:

- Speicherung nach dem Mehrheitsprinzip (siehe [Quorum](#))
- Unterstützung von zwei Technologiepfaden

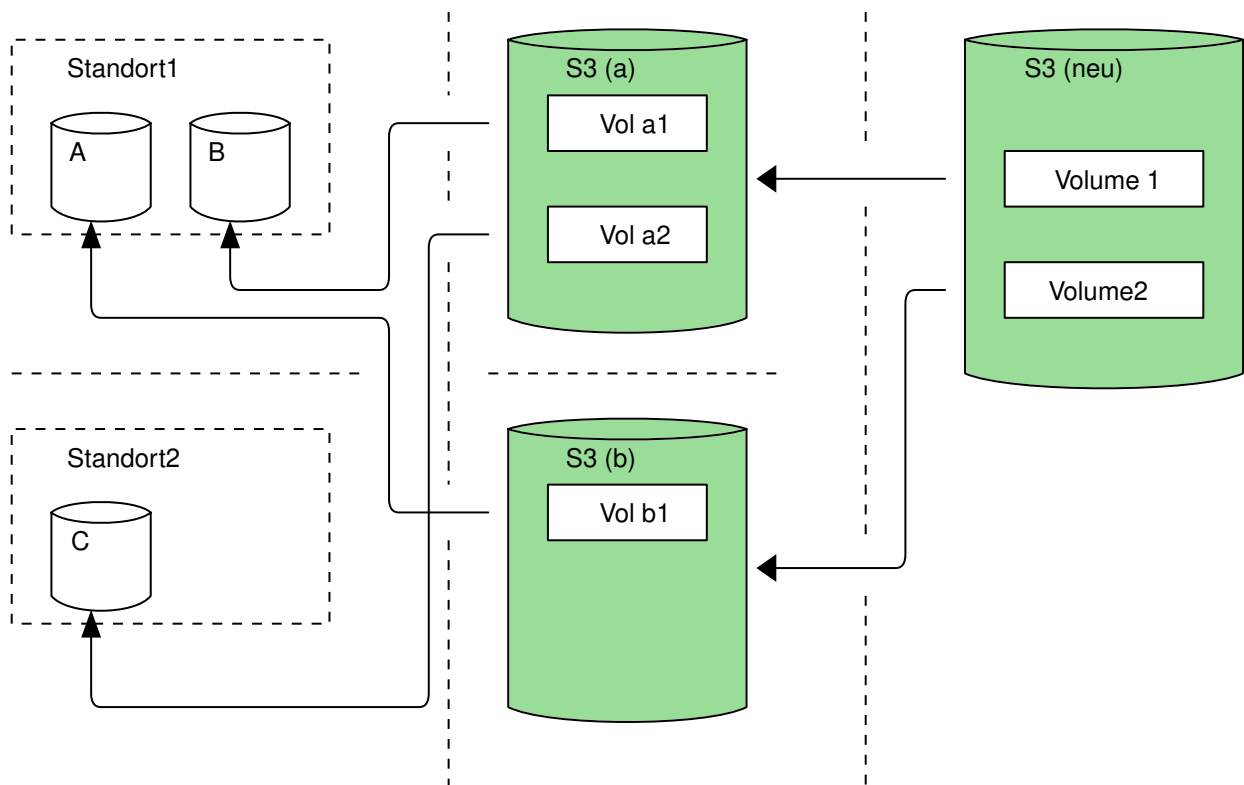


Abbildung 7. Speicherpfade, S3

## Anzahl Volumes

Die benötigte Anzahl an Volumes ergibt sich aus:

- Wieviele Volumes maximal gleichzeitig vergrößert werden sollen
- Welches Sicherheitsniveau währenddessen gehalten werden soll



Soll maximal ein Volume erweitert und währenddessen 3 Kopien weiterhin produktiv beschreibbar sein, dann braucht es (nach der Quorum Regel) insgesamt  $N=5$  Volumes, damit gilt:  $q=3=(N-1)/2+1$ .

# Fazit

- Um eine sichere Anbindung von Permanent-Speicher für Archivemata zu gewährleisten, müssen die Dateisysteme des ZIH möglichst direkt und nicht als network-attached storage herausgereicht werden.

- Für MinIO müssen im Vorfeld die Anzahl an Volumes festgelegt werden, da diese nicht änderbar ist.
- In MinIO wird ein Quorum anhand der Volumes gebildet
- Der Zustand eines MinIO wird im Storage abgelegt

[1] Die freie MinIO Implementierung wurde 2025 **eingeschränkt** [<https://blocksandfiles.com/2025/06/19/minio-removes-management-features-from-basic-community-edition-object-storage-code/>]. Zur Zeit ist nicht absehbar, ob es einen freien Klon geben wird

[2] MinIO ist unter der GNU AGPLv3 lizenziert, verfügbar unter <https://min.io/> und wird unter <https://github.com/minio/> entwickelt.

[3] Als URI, z.B. '<https://minio4.example.net>'

[4] Diese werden im Unterverzeichnis `.minio.sys` in **jedem** Volume abgelegt.

[5] Bei 3 Kopien gilt daher:  $N=3$ ,  $q=(N/2)+1 = 3/2+1 = 1+1 = 2$ , dh. zwei von drei Volumes müssen das Objekt speichern können, damit der Schreibvorgang erfolgreich ist.

[6] Quelle: <https://min.io/docs/minio/linux/operations/install-deploy-manage/deploy-minio-single-node-single-drive.html#id5>. Gilt auch für andere Konfigurationen.

[7] Im Fall von S3 wird der gemeinsame Namensraum durch die Buckets definiert

[8] Die Replikation wird im MinIO festgelegt und kann im Hintergrund erfolgen

[9] wie Server 'neu' in [Variante 2 mit separaten Mounts vom ZIH, Minio S3](#)

[10] Denkbar wäre, dass jede MinIO Instanz 3 Volumes enthält, die ersten beiden verweisen auf einen Storage, der die Daten der ersten Location abdeckt, der dritte wird als Mirror von der anderen MinIO Instanz verwendet, die über die zweite Location angesprochen wird, die als Replicator fungiert.