

cs-s3gateway

Handbuch

Nur lesender S3-Zugang und Web-Seite für ZFS-Snapshots und normale Ordner - ohne eigenes Datenformat

Version 0.3.1 · 13. Oktober 2026

Inhalt

1. Überblick	3
2. Schnellstart	3
3. Wie Ordner zu Buckets werden	3
4. Zugangsdaten und Zertifikate	4
5. Optionen und Konfigurationsdatei	4
6. Web-Seite	5
7. Clients	5
7.1 WinSCP (getestet)	5
7.2 rclone	5
7.3 aws cli und boto3	5
8. Unterstützte S3-Operationen	6
9. Sicherheitsmodell	6
10. Einbindung in napp-it CS	6
11. RustFS-Daten	7
12. Bauen, Tests, Grenzen	7
13. Lizenz	7

1. Überblick

cs-s3gateway stellt Ordner als S3-Buckets bereit, **ganz ohne eigenes Datenformat**: Der Objekt-Key ist der relative Pfad, die Dateien bleiben normale Dateien (SMB, **ls** und Backup-Tools sehen sie wie bisher). Der wichtigste Einsatz ist das Snapshot-Verzeichnis eines ZFS-Datasets: **Jeder Snapshot eines Datasets wird zu einem schreibgeschützten Ordner auf oberster Ebene eines S3-Buckets**, erreichbar mit WinSCP, rclone, restic, aws cli, boto3 - oder mit jedem Browser über die eingebaute Web-Seite.

```
cs-s3gateway --snaps s3-storage-snaps=/pool/dataset --bucket disaster-recovery=/pool/dr
```

`s3-storage-snaps/daily_2026-10-07/Projekte/x.docx` ist die Datei
`/pool/dataset/.zfs/snapshot/daily_2026-10-07/Projekte/x.docx`.

Warum ein eigenes Tool? RustFS und die meisten S3-Server halten ein eigenes Objekt-Layout und kennen weder einen schreibgeschützten Bucket noch "Snapshot als Bucket". Ein kleiner eigener Dienst, der nur *lesen* kann, ist einfacher zu prüfen als ein schreibfähiger Server, lässt sich bei Bedarf starten (mit Idle-Timeout) und hat keine Möglichkeit, Daten zu verändern.

- ein statisches Binary, nur Go-Standardbibliothek, keine Abhängigkeiten
- Windows, Linux (amd64, arm64), macOS (amd64, arm64), FreeBSD, illumos, Solaris
- nutzt Zugangsdaten und TLS-Zertifikat von RustFS in napp-it CS - nichts Neues einzurichten
- Menü **System > Services > cs-s3gateway** in napp-it CS: Start, Stop, Autostart, Idle-Timeout, Buckets

2. Schnellstart

In napp-it CS (Zugangsdaten und PEM-Dateien existieren schon von RustFS):

```
cs-s3gateway --snaps s3-storage-snaps=/tank/s3_storage --listen :9100 --idle-timeout 30m
```

Eigenständig, ohne TLS und mit expliziten Zugangsdaten (Test):

```
set CS_S3GW_ACCESS_KEY=s3admin
set CS_S3GW_SECRET_KEY=0123456789abcdefghij
cs-s3gateway --no-tls --listen 127.0.0.1:9100 --bucket docs=D:\share\docs
```

<http://127.0.0.1:9100/> im Browser öffnen oder einen S3-Client auf <http://127.0.0.1:9100> richten (Path-Style).

3. Wie Ordner zu Buckets werden

Option	Exportierter Ordner	Bucket-Inhalt
<code>--bucket name=/ordner</code>	<code>/ordner</code>	der Ordner selbst; Key = relativer Pfad
<code>--snaps name=/pool/dataset</code>	<code>/pool/dataset/.zfs/snapshot</code>	je Snapshot ein Ordner auf oberster Ebene

- Beide Optionen sind wiederholbar - mehrere Buckets auf mehreren Ordnern sind möglich.
- **Bucket-Namen** bestehen aus 3-63 Zeichen **a-z 0-9 -** und beginnen und enden mit Buchstabe oder Ziffer. **Objekt-Keys** sind beliebiges UTF-8 (Umlaute, Leerzeichen, CJK ...); sie sind einfach der Dateipfad mit `/` als Trenner.
- `.zfs` gehört zu **einem Dataset**. Snapshots von Child-Datasets liegen in deren eigenem `.zfs`: Für jedes Dataset, das erreichbar sein soll, wird ein eigener `--snaps`-Bucket angelegt. Der Pfad ist das *gemountete* Dateisystem (`zfs get mountpoint`), unter Windows zum Beispiel `D:/s3_storage`.
- Sichtbar sind nur normale Dateien und Ordner. Symlinks, Fifos, Geräte und Sockets gibt es für das Gateway nicht. Namen, die sich nicht sicher adressieren lassen (Backslash, Steuerzeichen, ungültiges UTF-8), werden ausgeblendet. Bei `--bucket` ist ein `.zfs` auf oberster Ebene versteckt.
- ZFS mountet Snapshots beim ersten Zugriff automatisch, der erste Zugriff in einen Snapshot kann deshalb langsamer sein.

4. Zugangsdaten und Zertifikate

Standardmäßig werden sie aus napp-it CS genauso übernommen wie vom RustFS-Startskript:

- Access Key **s3admin**, Secret = die ersten 20 Bytes der ersten Zeile von `<base>/_cfg/server.auth` (das RustFS-Login), oder ein eigenes **Session-Secret**: `--creds-from <datei> --access-key s3session` mit einer Datei, die ein zufälliges, nur dem Gateway bekanntes Secret enthält. Das napp-it CS Menü erzeugt es auf Wunsch (Schaltfläche *new secret*, Einstellung *Login*), legt es in `_cfg/s3gw/s3gw.secret` ab und zeigt es im Login-Fenster. Ein neues Secret widerruft das alte und startet den Dienst neu. Die Service-Tabelle zeigt den Access Key und für beide Logins eine Spalte *Secret*; Secrets werden nie in der Tabelle ausgegeben, *show* öffnet sie in einem Popup.
- Zertifikat und Schlüssel = `<base>/_cfg/s3/pem/rustfs_cert.pem` und `rustfs_key.pem`

Ein S3-Client, der mit RustFS funktioniert, funktioniert also mit denselben Schlüsseln auch mit dem Gateway - nur der Port ist ein anderer (RustFS 9000/9001, Gateway 9100).

Einstellung	Standard	Überschreiben
napp-it CS Ordner	<code>/opt/csweb-gui</code> (C:\opt\csweb-gui unter Windows)	<code>--base</code>
Secret-Datei	<code><base>/_cfg/server.auth</code>	<code>--creds-from datei</code>
Access Key	<code>s3admin</code>	<code>--access-key</code> , Umgebungsvariable <code>CS_S3GW_ACCESS_KEY</code>
Secret	aus der Secret-Datei	Umgebungsvariable <code>CS_S3GW_SECRET_KEY</code> (bleibt aus der Prozessliste)
Zertifikat / Schlüssel	<code><base>/_cfg/s3/pem/...</code>	<code>--cert, --key</code>

5. Optionen und Konfigurationsdatei

Option	Bedeutung
<code>--bucket name=ordner</code>	einen Ordner exportieren, wiederholbar
<code>--snaps name=mountpoint</code>	<code><mountpoint>/zfs/snapshot</code> exportieren, wiederholbar
<code>--rust name=ordner, --rustsnaps name=mountpoint</code>	RustFS-Datenordner, siehe Kapitel 11
<code>--listen :9100</code>	Listen-Adresse (Standard <code>:9100</code>)
<code>--idle-timeout 30m</code>	beenden nach dieser Zeit ohne Anfragen (<code>0</code> = bis zum Stoppen laufen)
<code>--log datei</code>	Logzeilen zusätzlich zu stdout an eine Datei anhängen
<code>--max-keys 1000</code>	maximale Zahl Keys je Listen-Antwort
<code>--no-tls</code>	reines HTTP (Test, oder hinter einem TLS-Proxy)
<code>--base, --creds-from, --access-key, --cert, --key</code>	Zugangsdaten und Zertifikat, siehe Kapitel 4
<code>--conf datei</code>	Optionen aus einer Datei lesen
<code>--version</code>	Version ausgeben

`--conf datei` liest Zeilen der Form `option=wert` (`#` beginnt einen Kommentar); Optionen auf der Kommandozeile haben Vorrang vor der Datei. napp-it CS schreibt diese Datei aus seinem Menü. Der Wert von `snaps=` und `bucket=` ist `name=pfad`, am ersten `=` geteilt. Secrets werden nie in dieser Datei gespeichert.

```
# s3gw.conf
listen=:9100
idle-timeout=30m
```

```
snaps=s3-storage-snaps=/tank/s3_storage
bucket=disaster-recovery=/tank/dr
```

6. Web-Seite

Das Gateway hat eine kleine eingebaute, nur lesende Web-Seite für den Zugriff "unterwegs" mit jedem Browser oder Handy. Man öffnet die Adresse des Gateways (<https://host:9100/>, sie leitet auf [/_browse/](#) weiter) und meldet sich mit demselben Access Key und Secret an. Man erhält die Bucket-Liste, Ordnerlisten (500 Einträge pro Seite) und zu jeder Datei:

- ein Klick auf den Namen zeigt `png jpeg gif webp pdf txt` direkt im Browser an, alles andere ist ein Download;
- der Link **Download** erzwingt immer einen Download (`?dl=1, Content-Disposition: attachment`).

HTML, SVG und alle anderen aktiven Inhaltstypen werden **nicht dargestellt**; jede Datei-Antwort trägt `Content-Security-Policy: sandbox` und `X-Content-Type-Options: nosniff`, und Seiten sind `no-store`. Die Seite nutzt HTTP-Basic-Authentifizierung, sie ist deshalb **nur über HTTPS** zu verwenden (Standard); der Begrenzer für fehlgeschlagene Logins der S3-Seite gilt ebenfalls. Der Namensraum `/_browse` kann nicht mit einem Bucket kollidieren, weil Bucket-Namen kein `_` enthalten dürfen.

Die eigene Web-Konsole von RustFS kann einen fremden S3-Endpunkt nicht anzeigen; deshalb bringt das Gateway eine eigene Seite mit.

7. Clients

Unterstützt wird nur die Path-Style-Adressierung (<https://host:9100/bucket/key>), Region `us-east-1`. Bei einem selbstsignierten Zertifikat wird es importiert oder die Prüfung im Client abgeschaltet.

7.1 WinSCP (getestet)

Dateiprotokoll *Amazon S3*, Hostname und Port 9100, Access Key `s3admin` und Secret. Unter Erweitert > Umgebung > S3 den URL-Stil auf **Pfad** stellen (Pflicht: WinSCP nutzt standardmäßig den Virtual-Host-Stil, den das Gateway nicht unterstützt). Bei einem selbstsignierten Zertifikat *TLS-Zertifikat prüfen* abwählen.

7.2 rclone

```
[snaps]
type = s3
provider = Other
endpoint = https://host:9100
access_key_id = s3admin
secret_access_key = <secret>
force_path_style = true
region = us-east-1
no_check_bucket = true

rclone ls snaps:s3-storage-snaps/daily_2026-10-07
rclone copy snaps:s3-storage-snaps/daily_2026-10-07/Projekte /restore
```

7.3 aws cli und boto3

```
aws s3 ls s3://s3-storage-snaps/ --endpoint-url https://host:9100 --no-verify-ssl
aws s3 cp s3://s3-storage-snaps/daily_2026-10-07/x.docx . --endpoint-url https://host:9100 --no-verify-ssl

import boto3
from botocore.config import Config
s3 = boto3.client("s3", endpoint_url="https://host:9100", aws_access_key_id="s3admin",
                  aws_secret_access_key="...", region_name="us-east-1", verify=False,
                  config=Config(s3={"addressing_style": "path"}))
for o in s3.list_objects_v2(Bucket="s3-storage-snaps", Delimiter="/").get("CommonPrefixes", []):
    print(o["Prefix"])
```

Presigned URLs (`aws s3 presign, generate_presigned_url`) funktionieren für GET und HEAD.

8. Unterstützte S3-Operationen

Operation	Status
ListBuckets	ja
ListObjects (v1) / ListObjectsV2	ja - prefix, delimiter, max-keys, continuation, <code>encoding-type=url</code>
GetObject / HeadObject	ja - <code>Range</code> und bedingte Header (<code>If-Modified-Since, ...</code>)
GetBucketLocation, GetBucketVersioning	ja (immer <code>us-east-1</code> , Versionierung nicht aktiv)
presigned GET/HEAD (SigV4)	ja
andere Bucket-Unterressourcen (ACL, Policy, Tagging, ...)	<code>501 NotImplemented</code>
PUT, POST, DELETE, Multipart-Upload, Copy	<code>AccessDenied</code> - "cs-s3gateway is read-only"

ETags werden aus Größe und Änderungszeit abgeleitet, nicht aus einer MD5-Summe des Inhalts. Das genügt für Caching und für Sync-Tools, die Größe und Zeit vergleichen; Tools, die MD5-ETags verlangen, können ihre `--size-only`-artigen Optionen nutzen.

9. Sicherheitsmodell

- **Nur lesend.** Implementiert sind nur GET, HEAD und Auflisten; jede andere Anfrage wird mit `AccessDenied` beantwortet. Kein Codepfad schreibt in die exportierten Ordner, und ZFS-Snapshots sind zusätzlich unveränderlich.
- **Ordner-jail.** Jeder Dateizugriff läuft über Gos `os.Root`: weder `..` noch ein Symlink kann den exportierten Ordner verlassen. Es werden nur normale Dateien und Ordner geöffnet, ein Fifo oder Gerät kann den Server also nie blockieren.
- **AWS Signature V4** für Header- und Presigned-Anfragen, Vergleich in konstanter Zeit, 15 Minuten Zeitabweichung erlaubt.
- **Brute-Force-Begrenzer.** 10 fehlgeschlagene Logins innerhalb einer Minute sperren die Client-Adresse für eine Minute (S3 und Web-Seite).
- **TLS** (mindestens 1.2) mit dem RustFS-Zertifikat; reines HTTP nur mit `--no-tls`.
- **Härtung der Web-Seite.** Basic-Auth über TLS, strenge CSP, `nosniff`, `sandbox` bei Datei-Antworten, HTML-maskierte Namen, aktive Inhaltstypen nur als Download, Anfragepfade in Logzeilen in Anführungszeichen.
- **Minimale Rechte.** Den Dienst als unprivilegierten Benutzer betreiben, der die exportierten Ordner und die `_cfg`-Dateien lesen kann und sonst nichts. Das Secret wird weder geloggt noch in der Konfigurationsdatei gespeichert.
- **Geprüft.** Befunde, Korrekturen und bekannte Einschränkungen stehen in SECURITY.md. Buckets, die den napp-it `_cfg`-Ordner (server.auth, TLS-Schlüssel) freilegen würden, werden abgelehnt; `.zfs` bleibt unter Windows und macOS unter jedem Alias versteckt.
- **Idle-Timeout.** `--idle-timeout` beendet den Dienst von selbst, ein Dienst auf Abruf ist also nicht den ganzen Tag ein offener Port.
- **Erreichbarkeit.** Das Gateway ist für ein vertrauenswürdiges LAN oder VPN gedacht, oder für den Betrieb hinter einem Reverse-Proxy. Wer es im Internet veröffentlicht, nutzt wie bei jeder Admin-Oberfläche HTTPS und eine Firewall- oder IP-Beschränkung - es verwendet die Admin-Zugangsdaten von RustFS (noch keine Benutzerkonten).

10. Einbindung in napp-it CS

napp-it CS enthält das Menü **System > Services > cs-s3gateway**. Es

- startet und stoppt den Dienst und schaltet den **Autostart** ein und aus (Start mit `server.pl` und beim Boot),
- stellt Port, **Idle-Timeout** (nie, 15 Min ... 1 Tag) und HTTPS ja/nein ein,
- verwaltet die Bucket-Liste (`snap:name=/pool/dataset`, `dir:name=/ordner`; eine leere Liste wird mit den Snapshots des RustFS-Dateisystems `s3_storage` vorbelegt),
- zeigt den Endpunkt, einen Link zur Web-Seite und ein **Login**-Popup mit Access Key und Secret,
- schreibt `_cfg/s3gw/s3gw.conf` und legt Pid-Datei und Log daneben ab.

Das Binary wird in `_my/tools/cs-s3gateway/<os>.amd64/` erwartet; es wird über das Menü **About > Download cs-tools** verteilt.

11. RustFS-Daten

RustFS hält seine Objekte in einem eigenen Format (je Objekt ein Ordner mit `xl.meta` plus Teildateien), nicht als normale Dateien. Zwei Bucket-Typen lesen dieses Format direkt und zeigen die Objekte als normale Dateien:

```
cs-s3gateway --rust name=/s3_storage           # RustFS-Datenordner: <rustfs-bucket>/<objekt>
cs-s3gateway --rustsnaps name=/tank/s3_storage # <dataset>/zfs/snapshot/<snapshot>/<rustfs-
        bucket>/<objekt>
```

Im napp-it CS Menü werden die Bucket-Zeilentypen `rust:name=pfad` und `rustsnap:name=pfad` verwendet, zum Beispiel `rustsnap:s3-storage-snaps=d:/s3_storage`.

Unterstützt: RustFS mit einer Platte (`xl.meta` v2, Inline-Daten, Multipart-Objekte), verschachtelte Keys, die neueste Version versionierter Objekte (Delete-Marker verbergen das Objekt), echter ETag und Content-Type aus den Metadaten. Die internen Ordner (`.rustfs.sys`, `xl.meta`, Teildateien) sind nie sichtbar oder adressierbar.

Nicht unterstützt (werden gelistet, GET antwortet aber mit 501 und Begründung): komprimierte oder verschlüsselte Objekte, Tiering-Objekte und Erasure-Coding-Daten über mehrere Platten. Die Bitrot-Prüfsummen der Teildateien werden übersprungen, nicht geprüft. Ältere Versionen eines Objekts werden nicht ausgeliefert.

12. Bauen, Tests, Grenzen

Verwendet wird Go 1.26 (der Ordner-Jail braucht `os.Root`, Go 1.24 oder neuer).

```
go vet ./... && go test ./...
powershell -NoProfile -File build-all.ps1 -Test
```

`build-all.ps1` erzeugt 8 statische Varianten (`CGO_ENABLED=0`, `-trimpath -s -w`) in `dist\release` plus `checksums.sha256`. Die Tests decken die dokumentierten AWS-Signature-V4-Beispiele (Header und Presigned), einen vollständigen List/Get/Range/Presign-Durchlauf einschließlich Unicode-Namen, Paging und Ordner-Jail, den Begrenzer für fehlgeschlagene Logins, die Prüfung von Bucket-Namen und die Web-Seite (Login, HTML-Maskierung, Download-Header, Traversal-Versuche) ab.

Grenzen: nur lesend by Design - keine Uploads, kein Löschen, kein Multipart, keine Versionierung, keine Benutzerkonten oder Bucket-Policies; ein Satz Zugangsdaten (die Admin-Schlüssel von RustFS); nur Path-Style-Adressierung (keine `bucket.host`-Virtual-Hosts). Getestet mit der Go-Testsuite, boto3, WinSCP (URL-Stil Pfad) und der Browser-Seite unter Windows; Symlink-Behandlung und das Auto-Mount-Verhalten von ZFS-Snapshots unter Linux und illumos brauchen noch Feldtests.

13. Lizenz

BSD 2-Clause - siehe Datei LICENSE; Änderungen stehen in CHANGELOG.md.