



Honkio S3

Installation, deployment and operations guide

S3-compatible object storage that replaces MinIO: features, installation on a virtual machine and on Kubernetes, ports and reverse proxy, secrets and bucket encryption with OpenBao, web console, quotas, replication, compression, monitoring, troubleshooting and migration from MinIO.

Source code: <https://gitlab.honkio.net/honkio-oss/honkio-s3>

Version: commit 17ff020 · built 2026-10-22

This PDF is built from the repository's documents with `python docs/build-pdf.py`.

Contents

1. Overview

- Implemented
- Running
- Disk layout
- Tests
- Migration from MinIO

2. Installation and deployment

1. What you need
2. Getting the binary
3. Installation on a virtual machine (systemd)
4. Ports, firewall and reverse proxy
5. Secrets and OpenBao
6. First test
7. Monitoring
8. Upgrade, backup and restore
9. Migration from MinIO
10. Troubleshooting
11. Checklist before test use

3. Operations

1. Architecture in brief
2. Web console
3. Users and rights
4. Quotas
5. Per-bucket replication
6. Lifecycle and bitrot
7. All settings (environment variables)
8. Supported S3 operations

4. Migrating from MinIO

0. Quick migration: honkio-s3 migrate (recommended)
1. Phases and schedule
2. Survey: what is in MinIO
3. Preparation
4. Users and policies
5. Bucket settings
6. Initial copy
7. Continuous sync
8. Verification and testing
9. Switch
10. After the switch
11. Rollback
12. Differences clients may notice

13. Checklist

Appendix A: systemd service

Appendix B: Settings file (no secrets)

Appendix C: OpenBao policy: reading secrets

Appendix D: OpenBao policy: bucket encryption keys

Appendix E: OpenBao Agent configuration

Appendix F: OpenBao Agent template for the secrets file

Appendix G: OpenBao Agent service

Appendix H: Kubernetes and External Secrets Operator

Appendix I: Dockerfile

Appendix J: Migration tool honkio-s3-cutover

Appendix K: Migration router (nginx)

Appendix L: Migration sync service

Appendix M: IAM sync service

Appendix N: IAM sync timer

Appendix O: Migration check script

Appendix P: Migration tool self-test

1. Overview

Source: README.md

Honkio S3 Server is an S3-compatible object storage server written in Rust. It replaces MinIO in Honkio's use (media.honkio.com), because MinIO has stopped developing its free version.

The scope is based on what Honkio projects actually use: honkio-server and dundee-server (python-minio + MinioAdmin), aws-sdk-js v3 projects, minio-js projects and the Dart minio package. Each Honkio S3 installation is a single server (no cluster), and data is distributed to other servers with per-bucket replication. Data is stored as plain files, so rsync/restic is enough for backups.

Implemented

Area	Support
Authentication	SigV4 header, presigned URL (GET/PUT, <code>response-*</code> overrides), aws-chunked streaming (signed and unsigned, CRC32/CRC32C trailers), <code>Content-MD5</code> and <code>x-amz-content-sha256</code> are verified. Any region is accepted.
Buckets	List, Create, Head, Delete, GetBucketLocation, bucket policy (Get/Put/Delete), versioning (always off), CORS/tagging (stored as is)
Objects	Put, Get (Range, If-* conditions), Head, Delete, DeleteObjects, CopyObject (COPY/REPLACE), object tagging, ListObjects V1/V2/Versions (prefix, delimiter, paging, <code>encoding-type=url</code>)
Multipart	Create, UploadPart, UploadPartCopy, Complete, Abort, ListParts, ListMultipartUploads. Abandoned uploads are cleaned up automatically (default 24 h).
Access control	Root credentials from environment variables, IAM users and named policies (Allow/Deny, wildcards, <code>s3:prefix</code> conditions), MinIO's canned policies, anonymous access through bucket policy (for example Honkio's <code>public/*</code>)
MinIO admin API	<code>add-user</code> (encrypted body), <code>remove-user</code> , <code>set-user-status</code> , <code>user-info</code> , <code>list-users</code> , <code>add/remove/info/list-canned-policy</code> , <code>set-user-or-group-policy</code> , <code>idp/builtin/policy/attach detach</code> , <code>info</code> . With these, honkio-server's <code>_auto_create_bucket</code> and <code>mc admin user/policy</code> work as is.
Other	<code>/minio/health/live ready cluster</code> (ready checks that the data disk is writable), CORS preflight (default <code>*</code> , wildcard origins such as <code>https://*.honkio.com</code>), SigV2 signatures for old clients, conditional writes (<code>If-None-Match: *</code>), graceful shutdown
Bitrot protection	A CRC32C is stored for each object at write time. A background scanner rereads every object once every <code>HONKIO_S3_SCRUB_INTERVAL_DAYS</code> (default 30 days, throttled by <code>HONKIO_S3_SCRUB_RATE_MB</code> , default 50 MiB/s). A corrupted object is not served (500 <code>XMinioObjectCorrupted</code>). The finding is written to the log, to metrics and to the file <code>.honkio-s3.sys/corrupt.jsonl</code> , and the object is restored from backup or uploaded again. Ideas from Swift's auditor, Garage's scrub and Ozone's scanner.
Metrics	Prometheus on MinIO's paths <code>/minio/v2/metrics/cluster bucket node</code> and <code>/metrics</code> : requests and latency histograms per API, error codes, traffic, bucket usage and quotas, disk space, scrub and lifecycle counters. The token is created with the <code>mc admin prometheus generate</code> command, or <code>MINIO_PROMETHEUS_AUTH_TYPE=public</code> .
Audit log	<code>HONKIO_S3_AUDIT_LOG=stdout</code> or a file path. One JSON line per request: time, request id, IP (X-Forwarded-For is trusted only from a proxy in a private network or in the <code>HONKIO_S3_TRUSTED_PROXIES</code> networks), access key / anonymous, API, bucket, key, status, error code, bytes and duration.

Area	Support
Quotas	Hard bucket quota: <code>mc admin bucket quota / MinioAdmin.bucket_quota_set</code> or the Quota button in the web console (the bucket list shows usage / quota). Enforced on PUT, multipart and copy.
Replication	Per-bucket one-way replication to one or more S3-compatible targets (another Honkio S3, MinIO, AWS S3 ...), configured from the web console (Replication). Every write, metadata or tag change and delete is forwarded to the target immediately (four work queues, order is kept for the same key, 3 attempts). Background reconciliation compares the listings of the bucket and the target, copies missing or outdated objects and deletes objects from the target that were deleted: at startup, after an error (backoff 2 min ... 30 min), on request and every <code>HONKIO_S3_REPLICATION_RESYNC_MINUTES</code> (default 360), throttled by <code>HONKIO_S3_REPLICATION_RESYNC_RATE_MB</code> (default 20 MiB/s). The target's version is recorded in the object's metadata, so the comparison does not depend on the target's clock or ETag style. Per target: filtering by key prefix, delete replication on/off (off = the copy survives accidental deletes), status and errors in the console and in Prometheus metrics (<code>honkio_s3_replication_*</code>). The target's secret key is not visible through the API, and it is stored in the file <code>.honkio-s3.sys/buckets/<bucket>/replication.json</code> (0600). Replication writes are marked (<code>x-honkio-s3-replication</code>), and a receiving Honkio S3 does not forward them.
Lifecycle	<code>PutBucketLifecycleConfiguration</code> : Expiration (Days/Date), filters Prefix, Tag and ObjectSize, <code>AbortIncompleteMultipartUpload</code> . The scanner runs the rules every <code>HONKIO_S3_SCAN_INTERVAL_MINUTES</code> (default 15). Transition rules are rejected.
Web console	Separate port (<code>--console-address / HONKIO_S3_CONSOLE_ADDRESS</code> , like MinIO's console), so HTML uploaded to buckets cannot run scripts in the console's origin. Login with access key and secret. Object browser: upload (files over 64 MiB as multipart), folder upload, drag & drop, download to your own machine, preview, delete (folders too), bucket creation and deletion, anonymous access setting (private / <code>public/*</code> / <code>public</code> / custom policy). Administration: users (create, policies, on/off, delete), policies (JSON editor), usage and disk space view. The session is an HMAC-signed <code>HttpOnly</code> + <code>SameSite=Strict</code> cookie (12 h) that is bound to the user's secret. Changing the secret or disabling the user ends the session immediately. All object operations go through the S3 handler with the same permission checks and audit log. Failed logins lock out the IP with increasing delays (5 attempts, then 30 s ... 15 min). <code>MINIO_BROWSER_REDIRECT_URL</code> redirects a browser that arrives at the S3 root to the console, and <code>MINIO_BROWSER=off</code> disables the console.
Disk compression	<code>HONKIO_S3_COMPRESSION=zstd</code> (default off): new objects are compressed on disk with <code>zstd</code> and decompressed on the fly when read. Clients see no difference: size, ETag, Range and presigned URLs work as before. The goal is the lowest possible CPU cost: level 1 (<code>HONKIO_S3_COMPRESSION_LEVEL</code>), already compressed formats (images, video, audio, zip, Office) are not compressed, the first 1 MiB block is tried and if the saving is under 1/8, the object is stored uncompressed. Objects under 4 KiB are not compressed. Files are in <code>zstd</code> seekable format (1 MiB frames + seek table), so a Range read decompresses only the frames it needs and <code>zstd -d</code> returns the original file. The bitrot check reads the compressed file without decompressing. The saving is shown on the console's Monitoring page and in the metric <code>honkio_s3_bucket_usage_stored_bytes</code> .
Bucket encryption	Per-bucket encryption at rest (Encryption in the console or <code>mc encrypt set sse-s3</code>): AES-256-GCM in 64 KiB segments, a separate key per bucket and a key derived from it per object. The bucket key is on disk only encrypted with an OpenBao Transit key, and the service decrypts it into memory at startup (<code>HONKIO_S3_OPENBAO_*</code>). If OpenBao does not respond, encrypted buckets return 503 until the key is available. Plaintext data is not written. Turning encryption on also encrypts existing objects in the background. Works with compression, multipart, copy, Range and replication. See <code>docs/en/OPERATIONS.md</code> and <code>docs/en/DEPLOYMENT.md</code> .
Public serving	<code>HONKIO_S3_PUBLIC_CACHE_CONTROL</code> (for example <code>public, max-age=86400</code>) for anonymous read requests when the object has no Cache-Control of its own. ETag, Last-Modified, 304 and Range for seeking in videos.

Not implemented: clustering and erasure coding (single-node server; fault tolerance comes from bucket replication to another server), versioning, bucket notifications, the per-object SSE header and SSE-C/SSE-KMS, object lock, MinIO's `mc replicate` API (replication is configured from the console), active-active replication, groups, STS and presigned POST. None of these were found in Honkio's code. A PUT of notification settings returns `NotImplemented`, so that a non-working feature is not silently accepted.

As in MinIO, bucket policy applies only to anonymous requests. Authenticated users are evaluated with IAM policies. Supported policy conditions are `s3:prefix`, `aws:SourceIp`, `aws:SecureTransport`, `aws:Referer` and `aws:UserAgent` (String*, IpAddress, Bool and Null operators). An unknown condition is rejected on save (`MalformedPolicy`), so a Deny rule cannot silently fail to take effect.

License: Honkio S3 does not contain MinIO (AGPLv3) code. MinIO was used only as a behavioral reference.

Running

Download: static binaries for Linux (x86_64, ARM64), macOS (Apple Silicon, Intel) and Windows from the [releases](#), and the container image `phonkonen/honkio-s3` (amd64, arm64) from Docker Hub.

Production installation (systemd or Kubernetes, ports, reverse proxy, OpenBao secrets): [docs/en/DEPLOYMENT.md](#).

```
MINIO_ROOT_USER=admin MINIO_ROOT_PASSWORD=a-long-password honkio-s3 server /data --address :9000
```

The environment variables use MinIO's names (`MINIO_ROOT_USER`, `MINIO_ROOT_PASSWORD`, `MINIO_ADDRESS`, `MINIO_REGION`, `MINIO_API_CORS_ALLOW_ORIGIN`). This is why existing manifests and `mc alias` work as is.

There is a CLI for offline maintenance: `honkio-s3 admin --data /data user add|remove|list, policy add|attach`. Migration from MinIO: `honkio-s3 migrate` (see below).

Disk layout

<code>/data/<bucket>/<key></code>	object data as a plain file
<code>/data/.honkio-s3.sys/meta/<bucket>/<key>.json</code>	ETag, content-type, x-amz-meta-*, tags, compression
<code>/data/.honkio-s3.sys/buckets/<bucket>/</code>	policy.json, cors.xml, ...
<code>/data/.honkio-s3.sys/multipart/<upload-id>/</code>	unfinished multipart uploads
<code>/data/.honkio-s3.sys/iam/iam.json</code>	users and policies (contains secret keys!)

In an encrypted bucket, the data file starts with the bytes `HS3E` and the metadata contains `"encryption": "aes-256-gcm"`. The bucket key is stored, encrypted by OpenBao, in the file `.honkio-s3.sys/buckets/<bucket>/encryption.json`.

With compression on, the data file can be in `zstd` format. The metadata then contains `"compression": "zstd"` and `stored_size`, and you get the original with the command `zstd -d < file > copy`.

Writes are atomic: data first goes to a temporary file, the file is fsynced and only then renamed, and writes to the same key are serialized with a lock. The limitation is the same as in MinIO's old FS mode: the keys `a` and `a/b` cannot exist at the same time (`XMinioParentIsObject` / `XMinioObjectExistsAsDirectory`).

Tests

```
cargo test # unit tests, including AWS's SigV4 test vectors
python tests/compat/python_compat.py http://127.0.0.1:9000 USER PASS
```

```
# Honkio S3 features: server running with HONKIO_S3_SCAN_INTERVAL_MINUTES=1 HONKIO_S3_AUDIT_LOG=<file>
python tests/compat/features_test.py http://127.0.0.1:9000 USER PASS <DATA_DIR> <AUDIT_LOG>
# Disk compression: server running with HONKIO_S3_COMPRESSION=zstd
python tests/compat/compression_test.py http://127.0.0.1:9000 USER PASS <DATA_DIR>
# Bucket encryption: server running with OpenBao settings (HONKIO_S3_OPENBAO_ADDR + Login)
python tests/compat/encryption_test.py http://127.0.0.1:9000 USER PASS <DATA_DIR>
# Replication + console quota: source with console and a second server as target
python tests/compat/replication_test.py http://127.0.0.1:9101 http://127.0.0.1:9102 http://127.0.0.1:9201
USER PASS
# Migration from a real MinIO (empty MinIO and empty Honkio S3 directory; the test also runs Honkio S3)
python tests/compat/migrate_test.py http://127.0.0.1:9380 ROOT PASS ./target/release/honkio-s3
<EMPTY_DATA_DIR>
node tests/compat/node_compat.mjs http://127.0.0.1:9000 USER PASS <dir with @aws-sdk/client-s3> <dir with
minio>
```

The compatibility tests go through the flows of honkio-server's `minio_service.py` and `admin/applications.py`, presigned uploads and anonymous `public/*` access. They also cover dualimpact's metadata rewrite and rename, appmill's public bucket, and pagination and multipart in boto3 and aws-sdk.

Migration from MinIO

Full guide: [docs/en/MIGRATION.md](https://docs.honkio.com/en/MIGRATION.md). Honkio's own migration (media41, 65 buckets) was completed on 20.10.2026.

```
export MIGRATE_SOURCE_ACCESS_KEY=... MIGRATE_SOURCE_SECRET_KEY=... # MinIO root/admin, not on the
command line
honkio-s3 migrate --data /var/lib/honkio-s3 --source http://minio:9000 --dry-run # what moves, what to
check by hand
honkio-s3 migrate --data /var/lib/honkio-s3 --source http://minio:9000 --rate-mb 20
# at cutover: write freeze, repeat run (changes only) and proxy/DNS to Honkio S3
honkio-s3 migrate --data /var/lib/honkio-s3 --source http://minio:9000 --delete
```

`honkio-s3 migrate` migrates users, service accounts and policies with the same keys (group policies are attached to the members), buckets with their creation dates and settings (policy, CORS, lifecycle, tags, quota), and objects so that the key, ETag (multipart too), Last-Modified, Content-Type, headers, `x-amz-meta-*` and tags are kept. Data is written directly to the data directory, which keeps the ETag and timestamp. A repeat run copies only the changes. Features that Honkio S3 does not have (versioning, object lock, SSE, MinIO replication, notifications) are reported at the end of the run.

2. Installation and deployment

Source: docs/en/DEPLOYMENT.md

This guide is for a team that installs Honkio S3 in its own environment for testing. The guide covers:

- installation on a virtual machine (systemd) or in Kubernetes
- ports, firewall and reverse proxy settings
- secrets management through OpenBao.

Honkio S3 is an S3-compatible storage server built to replace MinIO. It supports MinIO's environment variables, the `mc` command and the MinIO admin API. The features are listed in README.md.

Distribution. Each Honkio S3 installation is a single process with a single data directory. Installations do not form a cluster where several machines share the same namespace and take over for each other automatically. Data is distributed with **per-bucket replication**: the changes in each bucket are copied within seconds to one or more other servers, for example another Honkio S3 in another data center or a cloud S3, and background reconciliation fixes differences that arose during outages. If the primary server fails, clients are moved to the copy with DNS or a proxy. The switch is done by hand, not automatically. Replication is described in more detail in OPERATIONS.md, chapter 5.

1. What you need

Operating system	Linux x86_64. The binary is statically linked (musl), so the distribution does not matter. Tested on Debian 12.
CPU and memory	1 vCPU and 512 MiB are enough for testing. Memory use is typically 50–150 MiB.
Disk	A local file system, for example ext4 or xfs. Objects are stored as ordinary files, and writes use the temporary file + fsync + rename pattern. Do not use NFS or any other network disk , and do not let two Honkio S3 processes use the same data directory.
TLS	Honkio S3 does not terminate TLS itself. You need a reverse proxy in front (nginx, Envoy, HAProxy, Ingress) that handles HTTPS.
Clients	Clients must use path-style addresses (<code>https://s3.example.com/bucket/key</code>). Virtual-host style (<code>bucket.s3.example.com</code>) is not supported. MinIO clients and Honkio services already use path style.
Tools	<code>mc</code> (MinIO Client) for testing and administration. Also <code>bao</code> (OpenBao CLI) if secrets are fetched from OpenBao.

2. Getting the binary

Choose one of the following methods.

a) Prebuilt binary (recommended). Every version is released as static binaries that need no other libraries. Download the latest version:

Platform	File
Linux x86_64	honkio-s3-linux-x86_64
Linux ARM64 (e.g. Graviton, Raspberry Pi 4/5 64-bit)	honkio-s3-linux-arm64

Platform	File
macOS Apple Silicon	<code>honkio-s3-macos-arm64</code>
macOS Intel	<code>honkio-s3-macos-x86_64</code>
Windows x86_64	<code>honkio-s3-windows-x86_64.exe</code>
Checksums	<code>SHA256SUMS</code>

All versions: <https://gitlab.honkio.net/honkio-oss/honkio-s3/-/releases>

```
curl -fLO https://gitlab.honkio.net/honkio-oss/honkio-s3/-/releases/permalink/latest/downloads/honkio-s3-linux-x86_64
curl -fLO https://gitlab.honkio.net/honkio-oss/honkio-s3/-/releases/permalink/latest/downloads/SHA256SUMS
sha256sum --ignore-missing -c SHA256SUMS
sudo install -m 0755 honkio-s3-linux-x86_64 /usr/local/bin/honkio-s3
```

Linux is recommended for production. The macOS and Windows binaries are meant for development and trying it out. macOS quarantines a file downloaded with a browser: remove the flag with `xattr -d com.apple.quarantine honkio-s3-macos-arm64`, or download the file with `curl`.

b) Container image from Docker Hub. The image is available for x86_64 and ARM64 hosts. It runs as uid 1000, the data directory is `/data` and the ports are 9000 (S3) and 9001 (console).

```
docker run -d --name honkio-s3 -p 9000:9000 -p 9001:9001 \
-e MINIO_ROOT_USER=admin -e MINIO_ROOT_PASSWORD="$(openssl rand -hex 24)" \
-e HONKIO_S3_CONSOLE_ADDRESS=:9001 \
-v honkio-s3-data:/data phonkonen/honkio-s3:latest
```

In production, pass the credentials as secrets (section 5) and pin the version with a tag, for example `phonkonen/honkio-s3:0.2.0`.

c) Build from source. You need Rust 1.99 or newer and a C compiler.

```
git clone https://gitlab.honkio.net/honkio-oss/honkio-s3.git && cd honkio-s3
cargo build --release --locked          # -> target/release/honkio-s3
cargo test                               # unit tests
```

Other platforms can be built on any machine with `cargo-zigbuild`, as the CI does (`.gitlab-ci.yml`, job `build:cross`). The repository's `Dockerfile` builds an image from source (`docker build -t honkio-s3 .`).

Finally, check that the binary works: `honkio-s3 --version`.

3. Installation on a virtual machine (systemd)

The repository contains ready-made files:

File	Target on the server
<code>deploy/systemd/honkio-s3.service</code>	<code>/etc/systemd/system/honkio-s3.service</code>
<code>deploy/systemd/honkio-s3.env.example</code>	<code>/etc/honkio-s3/honkio-s3.env</code> (settings, no secrets)
<code>deploy/openbao/*</code>	OpenBao Agent, see chapter 5

```
# User and directories
sudo useradd --system --home-dir /var/lib/honkio-s3 --shell /usr/sbin/nologin honkio-s3
sudo install -d -m 0750 -o honkio-s3 -g honkio-s3 /var/lib/honkio-s3
sudo install -d -m 0755 /etc/honkio-s3

# Binary
sudo install -m 0755 honkio-s3-linux-x86_64 /usr/local/bin/honkio-s3

# Settings: copy the example and edit the ports and addresses (chapter 4)
sudo install -m 0644 deploy/systemd/honkio-s3.env.example /etc/honkio-s3/honkio-s3.env
sudo install -m 0644 deploy/systemd/honkio-s3.service /etc/systemd/system/honkio-s3.service
```

Secrets go into the file `/etc/honkio-s3/secrets.env`, which OpenBao Agent writes (chapter 5). If you want to try without OpenBao first, you can create the file by hand. The password then does not appear on screen or in the shell history:

```
sudo sh -c 'umask 077; printf "MINIO_ROOT_USER=honkio-s3-admin\nMINIO_ROOT_PASSWORD=%s\n" "$(openssl rand -hex 24)" > /etc/honkio-s3/secrets.env'
```

Start the service:

```
sudo systemctl daemon-reload
sudo systemctl enable --now honkio-s3
systemctl status honkio-s3
curl -s -o /dev/null -w '%{http_code}\n' http://127.0.0.1:9000/minio/health/ready # 200
```

The root credentials have requirements: the user name must be at least 3 characters and the password at least 8. Otherwise the service does not start, and an error message appears in the log (`journalctl -u honkio-s3`).

The data directory is not given as an environment variable but on the `ExecStart` line (`honkio-s3 server /var/lib/honkio-s3`). If you want the data elsewhere, also change the `ReadWritePaths=` line.

4. Ports, firewall and reverse proxy

Ports

Service	Variable (command line)	Default	Open to
S3 API, health and metrics	<code>MINIO_ADDRESS (--address)</code>	<code>0.0.0.0:9000</code>	The reverse proxy, and if needed directly to internal network clients
Web console	<code>HONKIO_S3_CONSOLE_ADDRESS (--console-address)</code>	disabled	Only the reverse proxy. Restrict the console to a VPN or management network.

- The address can also be given as `:9000`, which means all addresses.
- If the proxy runs on the same machine, bind both ports to `127.0.0.1`. Then you do not need to open the firewall for them at all.
- `MINIO_BROWSER=off` disables the console completely.

The console runs **on its own port and its own hostname**, as in MinIO. The reason is security: HTML uploaded to buckets must not be able to run scripts in the console's origin. So do not publish the console as a subpath of the S3 hostname.

Firewall (when the proxy is on another machine)

```
# firewalld: only from the proxy's addresses
sudo firewall-cmd --permanent --add-rich-rule='rule family=ipv4 source address=10.0.0.10/32 port port=9000
protocol=tcp accept'
sudo firewall-cmd --permanent --add-rich-rule='rule family=ipv4 source address=10.0.0.10/32 port port=9001
protocol=tcp accept'
sudo firewall-cmd --reload

# ufw
sudo ufw allow from 10.0.0.10 to any port 9000 proto tcp
sudo ufw allow from 10.0.0.10 to any port 9001 proto tcp
```

Reverse proxy (nginx example)

```
# S3 API
server {
    listen 443 ssl http2;
    server_name s3.example.com;
    ssl_certificate /etc/ssl/s3.example.com/fullchain.pem;
    ssl_certificate_key /etc/ssl/s3.example.com/privkey.pem;

    client_max_body_size 0;           # no size limit (multipart parts can be large)
    proxy_request_buffering off;      # uploads straight to Honkio S3
    proxy_buffering off;             # downloads out without buffering
    proxy_read_timeout 3600s;
    proxy_send_timeout 3600s;

    location / {
        proxy_pass http://127.0.0.1:9000;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
        proxy_set_header Host $http_host;           # REQUIRED: Host is part of the SigV4 signature
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

# Web console: its own hostname, only from the management network
server {
    listen 443 ssl http2;
    server_name s3-console.example.com;
    ssl_certificate /etc/ssl/s3-console.example.com/fullchain.pem;
    ssl_certificate_key /etc/ssl/s3-console.example.com/privkey.pem;

    allow 10.0.0.0/8;                 # VPN and management networks
    deny all;

    client_max_body_size 0;
    proxy_request_buffering off;
    proxy_read_timeout 3600s;
    proxy_send_timeout 3600s;

    location / {
        proxy_pass http://127.0.0.1:9001;
        proxy_http_version 1.1;
        proxy_set_header Host $http_host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme; # this gives the session cookie the Secure flag
    }
}
```

The most important proxy settings:

- **Host is passed unchanged.** If the proxy changes the Host header, all signed requests fail with the error `SignatureDoesNotMatch`.
- **No size limit and no buffering.** Otherwise large uploads break off or fill the proxy's disk.
- **X-Forwarded-For and X-Forwarded-Proto.** The audit log, IP conditions in policies and the console's Secure cookie use them. Honkio S3 trusts the `X-Forwarded-For` header only if the request comes from a private or loopback address or from a `HONKIO_S3_TRUSTED_PROXIES` network. If the proxy comes from a public address, add its network to this variable, for example `HONKIO_S3_TRUSTED_PROXIES=203.0.113.0/24`.
- **MINIO_BROWSER_REDIRECT_URL.** Set it to the console's public address. When a browser opens the root of the S3 address, it is redirected to the console.

5. Secrets and OpenBao

What secrets Honkio S3 has

Secret	Where it comes from	Into OpenBao?
Root credentials (<code>MINIO_ROOT_USER</code> , <code>MINIO_ROOT_PASSWORD</code>)	Environment variables at startup	Yes: OpenBao is the only source, and the agent or ESO writes them for Honkio S3
IAM user keys (application credentials)	Honkio S3's own database <code><data>/ .honkio-s3.sys/iam/iam.json</code> , managed with <code>mc admin user</code> or from the console	Store the application's credentials in the application's own OpenBao path, where the application reads them. Honkio S3 does not fetch these from OpenBao.
Replication target keys	Entered in the console, stored in the file <code><data>/ .honkio-s3.sys/buckets/<bucket>/replication.json</code> (0600)	Keep the original in OpenBao (e.g. <code>secret/honkio-s3/replication/<target></code>) and enter it in the console. When rotating, update OpenBao first, then the console.
Keys of encrypted buckets	Honkio S3 creates them itself, and on disk they exist only encrypted with the OpenBao Transit key (<code><data>/ .honkio-s3.sys/buckets/<bucket>/encryption.json</code>)	The Transit key is in OpenBao and never leaves it. Honkio S3 only needs the <code>encrypt / decrypt</code> permission, see "Bucket encryption" below.
Console session key	Honkio S3 creates it itself on first start: <code><data>/ .honkio-s3.sys/console.key</code>	No. Deleting the key logs out all console users.

Because the data directory contains secrets (`iam.json` and `replication.json`), its **backups must be encrypted**, and the directory permissions are kept at 0750 (user `honkio-s3`).

Preparing OpenBao (once)

In the examples the KV v2 engine is mounted at the path `secret`. If you use a different path, also change it in the files `deploy/openbao/honkio-s3-read.hcl` and `secrets.env.ctmpl`.

```
export BAO_ADDR=https://openbao.example.com:8200
bao login -method=oidc # or your own method

# 1) Root credentials: the password is generated straight into OpenBao and is not shown on screen.
# Hex characters, so the password also works in a URL (MC_HOST_*).
openssl rand -hex 24 | tr -d '\n' | bao kv put secret/honkio-s3/root user=honkio-s3-admin password=-
```

```
# 2) Read access only to the Honkio S3 paths
bao policy write honkio-s3-read deploy/openbao/honkio-s3-read.hcl
```

Virtual machine: OpenBao Agent + AppRole

The agent logs in with AppRole and writes the credentials to the file `/etc/honkio-s3/secrets.env` (0600 root). Systemd reads the file as root before Honkio S3 starts as the user `honkio-s3`. When the secret changes in OpenBao, the agent rewrites the file and restarts Honkio S3.

```
# In OpenBao (once)
bao auth enable approle 2>/dev/null || true
bao write auth/approle/role/honkio-s3 token_policies=honkio-s3-read token_ttl=1h token_max_ttl=24h
secret_id_num_uses=0
```

First install the `bao` binary on the server at `/usr/bin/bao` from the OpenBao release page. Then:

```
sudo install -d -m 0700 /etc/openbao-agent/honkio-s3
sudo install -m 0600 deploy/openbao/agent.hcl /etc/openbao-agent/honkio-s3/agent.hcl
sudo install -m 0600 deploy/openbao/secrets.env.ctmpl /etc/openbao-agent/honkio-s3/secrets.env.ctmpl
sudo install -m 0644 deploy/openbao/openbao-agent-honkio-s3.service /etc/systemd/system/
sudoedit /etc/openbao-agent/honkio-s3/agent.hcl # vault.address (and ca_cert if you have your own CA)

# Role ID and Secret ID straight into files. Do not print them on screen.
bao read -field=role_id auth/approle/role/honkio-s3/role-id | sudo tee /etc/openbao-agent/honkio-s3/role_id >/dev/null
bao write -f -field=secret_id auth/approle/role/honkio-s3/secret-id | sudo tee /etc/openbao-agent/honkio-s3/secret_id >/dev/null
sudo chmod 0600 /etc/openbao-agent/honkio-s3/role_id /etc/openbao-agent/honkio-s3/secret_id

sudo systemctl daemon-reload
sudo systemctl enable --now openbao-agent-honkio-s3
sudo test -s /etc/honkio-s3/secrets.env && echo "secrets rendered"
sudo systemctl restart honkio-s3
```

You can run the `bao write ... secret-id` command from your own machine, as long as the pipe carries the result to the server. For example: `ssh host 'sudo tee ../secret_id >/dev/null'`. The Secret ID must not be copied into a chat, a ticket or a log.

Rotating the root password:

1. Run `openssl rand -hex 24 | tr -d '\n' | bao kv patch secret/honkio-s3/root password=-.`
2. The agent notices the change within five minutes, rewrites `secrets.env` and restarts Honkio S3.
3. At the same time, update every place where the root credentials are used. For example, the `mc alias` and the Prometheus token must be regenerated. Applications do not use the root credentials but their own IAM users.

Bucket encryption: OpenBao Transit

Encryption is per bucket (console → Encryption or `mc encrypt set sse-s3`). Honkio S3 logs in to OpenBao itself at startup and decrypts the bucket keys into memory. It only needs the `encrypt` and `decrypt` permission on the Transit key (`deploy/openbao/honkio-s3-transit.hcl`).

```
# In OpenBao (once)
bao secrets enable transit
bao write -f transit/keys/honkio-s3 # aes256-gcm96, not exportable, not deletable
bao policy write honkio-s3-transit deploy/openbao/honkio-s3-transit.hcl
# A separate AppRole for the service (transit permission only, short-lived tokens)
```

```
bao write auth/approle/role/honkio-s3-transit token_policies=honkio-s3-transit token_ttl=15m
token_max_ttl=1h
```

Virtual machine: put the credentials into files that only the service user can read. Do not print them on screen.

```
sudo install -d -m 0750 -o root -g honkio-s3 /etc/honkio-s3/openbao
bao read -field=role_id auth/approle/role/honkio-s3-transit/role-id | sudo tee /etc/honkio-
s3/openbao/role_id >/dev/null
bao write -f -field=secret_id auth/approle/role/honkio-s3-transit/secret-id | sudo tee /etc/honkio-
s3/openbao/secret_id >/dev/null
sudo chown honkio-s3 /etc/honkio-s3/openbao/role_id /etc/honkio-s3/openbao/secret_id
sudo chmod 0400 /etc/honkio-s3/openbao/role_id /etc/honkio-s3/openbao/secret_id
```

Add to the file `/etc/honkio-s3/honkio-s3.env`:

```
HONKIO_S3_OPENBAO_ADDR=https://openbao.example.com:8200
HONKIO_S3_OPENBAO_ROLE_ID_FILE=/etc/honkio-s3/openbao/role_id
HONKIO_S3_OPENBAO_SECRET_ID_FILE=/etc/honkio-s3/openbao/secret_id
#HONKIO_S3_OPENBAO_CACERT=/etc/honkio-s3/openbao/ca.pem
```

Kubernetes: use Kubernetes auth (`HONKIO_S3_OPENBAO_K8S_ROLE=honkio-s3`). The role is bound to the StatefulSet's ServiceAccount: `bao write auth/kubernetes/role/honkio-s3 bound_service_account_names=honkio-s3 bound_service_account_namespaces=storage token_policies=honkio-s3-transit ttl=15m`.

The startup log shows bucket encryption keys from OpenBao: ... and, for each encrypted bucket, encryption key of bucket X loaded from OpenBao. If OpenBao does not respond, encrypted buckets return 503 and the service retries in the background. Other buckets keep working.

Important: if the Transit key is lost, the data in encrypted buckets is lost. Make sure OpenBao is backed up. The key must not be deleted, and its `min_decryption_version` value must not be raised. Rotating the key (`bao write -f transit/keys/honkio-s3/rotate`) is safe: old versions still decrypt.

Kubernetes: External Secrets Operator

A ready-made example is in the file `k8s/honkio-s3-openbao.example.yaml`. It contains:

- a ServiceAccount, a SecretStore and an ExternalSecret
- a ConfigMap for the settings
- a StatefulSet (1 replica, PVC)
- a Service for both ports
- two Ingresses: S3 and a console restricted to the VPN.

The OpenBao Kubernetes auth role (once, when `auth/kubernetes` is already enabled):

```
bao write auth/kubernetes/role/eso-honkio-s3 \
  bound_service_account_names=eso-honkio-s3 \
  bound_service_account_namespaces=storage \
  token_policies=honkio-s3-read ttl=1h
```

Installation: in the example, edit the namespace, the OpenBao address, the image, the disk size and the hostnames. Then run `kubectl apply -f`. ESO refreshes the secret every 15 minutes. Honkio S3 reads the root credentials only at startup, so after a rotation run `kubectl rollout restart statefulset/honkio-s3 -n storage`.

6. First test

```
# The root credentials are read from OpenBao into environment variables. They do not end up in the shell
history or on screen.
export MC_HOST_mx="https://$(bao kv get -field=user secret/honkio-s3/root):$(bao kv get -field=password
secret/honkio-s3/root)@s3.example.com"
mc admin info mx
mc mb mx/testi
echo hello | mc pipe mx/testi/hello.txt
mc cat mx/testi/hello.txt

# A separate user for the application, with permissions only to its own bucket (same approach as in MinIO)
cat > /tmp/policy-testi.json <<'EOF'
{"Version":"2012-10-17","Statement":[{"Effect":"Allow","Action":["s3:*"],"Resource":
["arn:aws:s3:::testi","arn:aws:s3:::testi/*"]}]
EOF
mc admin policy create mx policy-testi /tmp/policy-testi.json
APP_SECRET=$(openssl rand -hex 20)
mc admin user add mx app-testi "$APP_SECRET"
mc admin policy attach mx policy-testi --user app-testi
printf %s "$APP_SECRET" | bao kv put secret/apps/testi/s3 access_key=app-testi secret_key=- # the
application's own path
unset APP_SECRET
```

Console: open <https://s3-console.example.com> and log in with the root credentials or with a user that has the `consoleAdmin` policy. In the console you can:

- browse objects, upload, preview and delete them
- set anonymous read access for a bucket (Access)
- set a quota (Quota)
- configure replication to another server (Replication)
- manage users and policies.

Compatibility tests: if you want to run the same tests as we do, the instructions are in the "Tests" section of README.md. The Python tests need the packages `minio`, `boto3` and `requests`.

7. Monitoring

Health	<code>GET /minio/health/live</code> returns 200 when the process is running. <code>GET /minio/health/ready</code> returns 200 when the data disk is writable, and 503 otherwise. The paths work without authentication.
Prometheus	The paths <code>/minio/v2/metrics/cluster</code> , <code>/minio/v2/metrics/node</code> , <code>/minio/v2/metrics/bucket</code> and <code>/metrics</code> on the S3 port. Create the token with <code>mc admin prometheus generate mx</code> , and store the <code>bearer_token</code> in OpenBao or in Prometheus's secrets. Another option is <code>MINIO_PROMETHEUS_AUTH_TYPE=public</code> if the port is visible only to the internal network. Useful alerts: <code>honkio_s3_scrub_corrupt_objects_total > 0</code> , <code>honkio_s3_replication_needs_resync == 1</code> for a long time, <code>minio_node_drive_free_bytes</code> and 5xx errors (<code>minio_s3_requests_5xx_errors_total</code>).
Logs	<code>journalctl -u honkio-s3</code> . <code>HONKIO_S3_AUDIT_LOG=stdout</code> writes a JSON line for every request with the IP, user, API, bucket, key, status and duration.
Bitrot	A background scanner verifies the checksum of every object once every 30 days. Findings go to the log, the metrics and the file <code><data>/honkio-s3.sys/corrupt.jsonl</code> . On a slow disk, lower the read rate with the setting <code>HONKIO_S3_SCRUB_RATE_MB=10</code> .

8. Upgrade, backup and restore

Upgrade. Replace the binary and restart the service. The data format stays compatible.

```
sudo install -m 0755 honkio-s3-linux-x86_64 /usr/local/bin/honkio-s3
sudo systemctl restart honkio-s3
```

In Kubernetes, change the image tag. Shutdown waits for running requests to finish (graceful shutdown).

Backup. Back up the whole data directory, including `.honkio-s3.sys`, which holds the metadata, IAM, bucket settings and the console key. For example `restic backup /var/lib/honkio-s3` or `rsync -a`. The files are ordinary files, so you can restore a single object by copying it into place. If the metadata file is missing, Honkio S3 builds the metadata from the file. Alongside or instead of backups you can use per-bucket replication to another server (console → Replication).

Restoring a whole machine.

1. Install the binary and settings as described in chapter 3.
2. Restore the data directory and set its owner to `honkio-s3:honkio-s3`.
3. Make sure OpenBao has the same root credentials.
4. Start the service.

Users and policies are restored with the data directory.

9. Migration from MinIO

The whole migration step by step (inventory, copy, verification, cutover and rollback) is described in `MIGRATION.md`. In short:

1. `honkio-s3 migrate --data <data> --source <minio-url> --dry-run` shows what will be migrated and what must be checked by hand. The source credentials are given in the environment variables `MIGRATE_SOURCE_ACCESS_KEY` and `MIGRATE_SOURCE_SECRET_KEY`.
2. The same command without the `--dry-run` option (e.g. `--rate-mb 20`) migrates the users with the same keys, the buckets with their settings (policy, CORS, lifecycle, tags, quota) and the objects so that the ETag, Last-Modified, metadata and tags are preserved.
3. At cutover: a write freeze in the proxy, a repeat run with `--delete` (copies only the changes), and the proxy or DNS pointed at Honkio S3.

10. Troubleshooting

Symptom	Cause and fix
<code>SignatureDoesNotMatch</code> on all requests	The proxy changes the Host header or the path. Use <code>proxy_set_header Host \$http_host;</code> and <code>proxy_pass</code> without a path.
<code>NoSuchBucket</code> or odd errors from a client	The client uses virtual-host style. Set path style, for example <code>addressing_style: path</code> in boto3 or <code>forcePathStyle: true</code> in AWS SDK v3.
Large uploads break off	The proxy's size limit or timeout. See the settings in chapter 4: <code>client_max_body_size 0</code> , buffering off and timeouts of 3600 s.

Symptom	Cause and fix
The service does not start: MINIO_ROOT_USER is not set	<code>secrets.env</code> is missing or empty. Check <code>systemctl status openbao-agent-honkio-s3</code> and the agent log (<code>journalctl -u openbao-agent-honkio-s3</code>): the AppRole credentials, the policy and the path <code>secret/honkio-s3/root</code> .
The console logs out immediately	The console is behind HTTP, but the browser receives <code>X-Forwarded-Proto: https</code> , or the other way round. Make sure the proxy sends the correct protocol.
Console: "Too many failed logins"	An IP address is locked for 30 seconds after five failed logins, and the lock grows to at most 15 minutes. If all users appear as the same IP, the proxy's <code>X-Forwarded-For</code> is not accepted: add the proxy's network to the <code>HONKIO_S3_TRUSTED_PROXIES</code> variable.
Replication status "Failing"	Open the bucket's Replication view and look at "Last error". The target must be reachable from the Honkio S3 machine. Note that many networks do not allow connections to their own public address (hairpin).
An encrypted bucket returns 503 ServiceUnavailable	The bucket's key could not be fetched from OpenBao. Check the log (<code>encryption key of bucket ... not available</code>): the address, the CA, the permissions of the AppRole files and the policy <code>honkio-s3-transit</code> . The key is fetched automatically when OpenBao responds.
<code>/minio/health/ready</code> returns 503	The data disk is full or read-only, or the permissions are wrong. Check <code>df -h</code> and the owner of the data directory.

11. Checklist before test use

- ☐ Binary installed, and `honkio-s3 --version` works.
- ☐ `/etc/honkio-s3/honkio-s3.env` : ports, `MINIO_BROWSER_REDIRECT_URL` and, if needed, `HONKIO_S3_TRUSTED_PROXIES` .
- ☐ Root credentials in OpenBao (`secret/honkio-s3/root`), the agent or ESO writes them, and the credentials are not in the repository, tickets or logs.
- ☐ The data directory is on a local disk, owner `honkio-s3` and permissions 0750.
- ☐ Reverse proxy: TLS, Host header, no size limit and no buffering, `X-Forwarded-*` headers.
- ☐ Console on its own hostname and restricted to the management network. The firewall opens the ports only to the proxy.
- ☐ `health/ready` returns 200, and `mc admin info` works.
- ☐ Prometheus token stored in the secrets, and scraping works.
- ☐ Encrypted backup of the data directory or replication to another server.
- ☐ If bucket encryption is used: Transit key `honkio-s3` and policy `honkio-s3-transit` in OpenBao, and OpenBao backups in order.

3. Operations

Source: docs/en/OPERATIONS.md

This guide complements the installation guide (DEPLOYMENT.md). It describes the web console, quotas, replication, access rights and all settings.

1. Architecture in brief

- **One process, one data directory.** Honkio S3 is a single Rust binary that listens on two ports: the S3 API and the web console. Objects are stored as plain files (`<data>/<bucket>/<avain>`). Metadata, users and settings are in the directory `<data>/honkio-s3.sys/`.
- **Writes are atomic.** Data is first written to a temporary file, the file is fsynced, and only then is it renamed into place. Writes to the same key are serialized with a lock. A reader therefore always sees either the old or the new version, never a partial one.
- **Background jobs:**
 - usage statistics and lifecycle expirations (every 15 min)
 - cleanup of incomplete multipart uploads
 - bitrot check (every object is read again every 30 days)
 - replication reconciliation.
- **Fault tolerance:** the server is not a cluster. A copy on another server is made with per-bucket replication (chapter 5), and the data directory is backed up.
- **Compatibility:**
 - Environment variables, the `mc` command, the MinIO admin API (users, policies, quotas), Prometheus paths and health paths work as in MinIO.
 - Tested with the python-minio, boto3, aws-sdk-js v3 and minio-js libraries and with `mc admin` commands.

<code><data>/<bucket>/<avain></code>	object data
<code><data>/honkio-s3.sys/meta/<bucket>/<avain>.json</code>	ETag, content-type, metadata, tags, checksum
<code><data>/honkio-s3.sys/buckets/<bucket>/replication.json, ...</code>	policy.json, quota.json, lifecycle.xml,
<code><data>/honkio-s3.sys/iam/iam.json</code>	users and policies (contains secret keys)
<code><data>/honkio-s3.sys/console.key</code>	console session key
<code><data>/honkio-s3.sys/multipart/</code>	incomplete multipart uploads
<code><data>/honkio-s3.sys/corrupt.jsonl</code>	bitrot findings

Overlapping key names follow the same rule as in MinIO's FS mode: `a` and `a/b` cannot exist at the same time.

2. Web console

The console opens at its own address, for example `https://s3-console.example.com`. If a browser opens the root of the S3 address, it is redirected to the console when `MINIO_BROWSER_REDIRECT_URL` is set.

Login. Log in with an access key and a secret key: the root credentials or an IAM user.

- A session lasts 12 hours. The cookie is signed and marked HttpOnly, SameSite=Strict and, behind HTTPS, Secure.
- The session ends immediately if the user's password is changed or the user is disabled.
- After five failed logins, the IP address is first locked for 30 seconds, and the lock grows to at most 15 minutes.

Views (shown according to the user's rights, see chapter 3):

View	Actions
Object Browser	Bucket list: object count, size and quota, anonymous access and a replication marker. Bucket creation and deletion (only an empty bucket can be deleted). Inside a bucket, folder browsing and search by key prefix. For objects: upload (files over 64 MiB as multipart), folder upload, drag and drop into the browser, download to your own machine, preview of images, video and audio, metadata display, and deletion, also of whole folders.
Access (anonymous access)	Private / public under <code>public/</code> (Honkio convention) / whole bucket public / custom bucket policy (JSON).
Quota	Hard quota in MiB, GiB or TiB units, or removal of the quota.
Replication	Replication targets, their status, "Resync now", editing and deletion (chapter 5). The Replication view in the sidebar collects the targets of all buckets.
Monitoring	Total buckets, objects and stored data, free disk space, and buckets in order of size with their quotas.
Users	User creation (the password can be generated), attaching policies, enabling and disabling a user, and deletion. You cannot delete or disable your own account.
Policies	Policy list and JSON editor. The built-in MinIO policies are shown, but they cannot be changed.

The console's object actions go through the same S3 handler as application calls. Rights, quotas and the audit log therefore apply to them in the same way. Content previewed in the browser is shown isolated (CSP sandbox), so HTML uploaded to a bucket cannot run a script in the console.

3. Users and rights

The rights model is the same as in MinIO. An application gets its own user and a policy that allows only its own buckets:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:*"],
      "Resource": ["arn:aws:s3:::sovellus-bucket", "arn:aws:s3:::sovellus-bucket/*"]
    }
  ]
}
```

- Authenticated requests are evaluated with the user's IAM policies. **A bucket policy applies only to anonymous requests**, as in MinIO. For example, Honkio's `public/*` read access is granted with a bucket policy.
- Policies support Allow and Deny rules, wildcards, and the conditions `s3:prefix`, `aws:SourceIp`, `aws:SecureTransport`, `aws:Referer` and `aws:UserAgent`. An unknown condition is rejected already when the policy is saved, so that a Deny rule does not silently fail to apply.
- The built-in policies are `readwrite`, `readonly`, `writelnonly`, `consoleAdmin` and `diagnostics`.

The console's admin views require the following admin rights (`consoleAdmin` includes all of them):

Action	Right
Listing, creating, deleting users and setting their status	<code>admin:ListUsers</code> , <code>admin:CreateUser</code> , <code>admin>DeleteUser</code> , <code>admin:EnableUser</code>
Listing, creating, deleting and attaching policies	<code>admin:ListUserPolicies</code> , <code>admin>CreatePolicy</code> , <code>admin>DeletePolicy</code> , <code>admin:AttachUserOrGroupPolicy</code>
Monitoring and usage data	<code>admin:ServerInfo</code>
Reading and setting quotas	<code>admin:GetBucketQuota</code> , <code>admin:SetBucketQuota</code>

Action	Right
Reading and setting replication	<code>admin:GetBucketTarget</code> , <code>admin:SetBucketTarget</code>

On the command line, the same actions are done like this: `mc admin user add|remove|enable|disable|info|list` , `mc admin policy create|remove|info|list|attach|detach` and `mc admin bucket quota` .

If the server is stopped, users can be managed with offline commands: `honkio-s3 admin --data <DATA> user add|remove|list` , `policy add|attach` and `import-minio-iam` .

4. Quotas

A bucket can have a **hard quota in bytes**. An upload that would take the bucket over the limit is rejected with the error `XMinioAdminBucketQuotaExceeded` . The check applies to plain uploads, multipart uploads and copies. The quota does not affect existing objects.

- **Console:** Object Browser → Quota (in the bucket list or inside a bucket). The bucket list and the Monitoring view show usage and quota as a bar, and the bar turns red when more than 90 % is in use.
- **Command line:** `mc admin bucket quota mx/bucket --size 10GiB` , `mc admin bucket quota mx/bucket (read)` and `mc admin bucket quota mx/bucket --clear` .
- **Application code:** python-minio's `MinioAdmin.bucket_quota_set()` works as is.
- **Metrics:** `minio_bucket_quota_total_bytes` and `minio_bucket_usage_total_bytes` .

Usage statistics are kept up to date on every write and delete, and the background scanner recalculates them in full. A quota cannot be set on the number of objects, and MinIO's old "fifo" type is not supported.

5. Per-bucket replication

Replication makes a **one-way copy** of a bucket (or part of it) on another S3-compatible server, for example another Honkio S3, MinIO, AWS S3 or another cloud S3. A bucket can have several targets.

How it works:

1. **Changes forwarded immediately.** Every write, metadata or tag change and delete is forwarded to the target within seconds. There are four work queues, and changes to the same key are always handled in the correct order. A failed transfer is retried three times (at 1 s and 5 s intervals), and after that the target is marked for reconciliation.
2. **Reconciliation (resync).** The listings of the bucket and the target are compared in key order. Missing and outdated objects are copied, and keys that were deleted from the source are deleted from the target (if delete replication is on). Reconciliation runs: - when a target is added - when the server starts - after an error, with backoff from 2 minutes to 30 minutes - from the console's "Resync now" button - every `HONKIO_S3_REPLICATION_RESYNC_MINUTES` (default 6 h).

Reconciliation copying is limited by the setting `HONKIO_S3_REPLICATION_RESYNC_RATE_MB` (default 20 MiB/s). Changes forwarded immediately are not limited. 3. **Bookkeeping.** The version each target has received is stored in the object's metadata. The comparison therefore does not depend on the target's clock or on the format of ETags (multipart). 4. **What is copied:** data, Content-Type, Cache-Control, Content-Disposition, Content-Encoding, Content-Language, Expires, `x-amz-meta-*` and tags. Large objects (over 64 MiB) are transferred as multipart and are not buffered in memory. 5. **Loops are prevented.** Replication writes are marked with the header `x-honkio-s3-replication` , and a receiving Honkio S3 does not forward them.

Adding a target in the console (Object Browser → bucket → Replication → Add target):

Field	Description
Endpoint URL	<code>https://s3.example.com</code> or <code>http://10.0.0.5:9000</code> : scheme, host and port only
Target bucket	The target bucket. Use a dedicated bucket for this copy.
Region	The target's region, default <code>us-east-1</code>
Access key / Secret key	Target credentials with rights to read, write and delete in the target bucket. When editing, an empty secret keeps the old one.
Only keys starting with	Limits replication to a key prefix, for example <code>public/</code> . Empty means the whole bucket.
Replicate deletes	On: what is deleted from the source is also deleted from the target. Off: the target keeps deleted objects, so the copy protects against accidental deletes.
Enabled	The target can be paused. When the target is enabled again, reconciliation runs.
Create the target bucket	Creates the target bucket if it does not exist

When you save, the connection to the target is checked. Invalid credentials and a missing bucket are reported immediately.

Statuses in the console:

Status	Meaning
In sync	Up to date, the latest reconciliation succeeded
Replicating	Changes in the queue
Catching up / Resyncing	Reconciliation is waiting or running
Failing	The target does not respond or rejects requests. "Last error" gives the reason.
Waiting for first sync	The target was just added
Disabled	The target is paused

Monitoring: metrics `honkio_s3_replication_pending`, `_objects_total`, `_bytes_total`, `_deletes_total`, `_failures_total`, `_needs_resync`, `_last_success_timestamp_seconds` and `_last_full_sync_timestamp_seconds` (labels `bucket` and `target`). Recommended alert: `honkio_s3_replication_needs_resync == 1` for more than an hour.

Notes:

- The target must be reachable from the Honkio S3 machine. In many networks a machine cannot connect to its own public address (hairpin NAT), so use the internal network address.
- The target's secret key is stored in the file `<data>/honkio-s3.sys/buckets/<bucket>/replication.json` (0600) and is not shown through the API. Keep the original key in OpenBao.
- Active-active replication between two servers (writes in both directions to the same bucket pair) is not supported. Changes forwarded immediately do not loop, but reconciliations would copy the same objects back and forth.
- MinIO's `mc replicate` command is not supported. Replication is set up from the console.

6. Lifecycle and bitrot

- **Lifecycle:** `mc ilm rule add` or `PutBucketLifecycleConfiguration`.
- Supported rules: Expiration (Days or Date), the filters Prefix, Tag and ObjectSize, and `AbortIncompleteMultipartUpload`.
- The scanner runs the rules every `HONKIO_S3_SCAN_INTERVAL_MINUTES`.
- Transition rules (moving to another storage class) are rejected.
- **Bitrot:**
- A CRC32C checksum is calculated for every object at write time.
- The background scanner reads the objects again at a throttled rate and compares the checksum.
- A corrupted object is not served. The request returns `500 xMinioObjectCorrupted`.
- The finding is recorded in the log, in the metric `honkio_s3_scrub_corrupt_objects_total` and in the file `corrupt.jsonl`.
- The object is restored from a backup or a replication target.

Disk compression

- Turn it on with the setting `HONKIO_S3_COMPRESSION=zstd` and a restart of the service. The default is off.
- Applies only to new writes (PUT, multipart, copy). Old objects stay uncompressed until they are written again.
- Compression is invisible to clients: size, ETag, Range and presigned URLs work as before.
- CPU cost is low: zstd level 1, already compressed formats (jpg, png, webp, mp4, mp3, zip, docx ...) are not compressed, and if the object's first 1 MiB shrinks by less than 1/8, the object is stored as is.
- The level can be tuned with the setting `HONKIO_S3_COMPRESSION_LEVEL`: negative = faster and less saving, 3–9 = more saving and more CPU.
- Compression can be turned off at any time. Compressed objects are still read correctly.
- **Do not go back to a binary older than the version that added compression** while compressed objects exist: the old version would serve compressed data for them.
- The saving is shown in the console (Monitoring → Stored data) and in the metric `honkio_s3_bucket_usage_stored_bytes` (compared with `minio_bucket_usage_total_bytes`).

Bucket encryption (OpenBao)

- Per-bucket setting: the console's **Encryption** button or `mc encrypt set sse-s3 <alias>/<bucket>` (S3: `PutBucketEncryption`, AES256). Off: from the console or `mc encrypt clear`.
- Object data is stored on disk with AES-256-GCM in 64 KiB segments (`HS3E` header). Every bucket has its own random 256-bit key. A separate key is derived from it for every object, so objects in the same bucket do not share a nonce space.
- The bucket key is on disk **only encrypted with the OpenBao Transit key** (`.honkio-s3.sys/buckets/<bucket>/encryption.json`). The service decrypts it into memory at startup (`transit/decrypt`). The Transit key never leaves OpenBao, so the disk or a backup alone is not enough to read the data.
- If OpenBao does not respond at startup, encrypted buckets respond 503 (`ServiceUnavailable`) until a background attempt gets the key (10 s, then less often, at most every 5 min). Plaintext data is never written to an encrypted bucket. Other buckets work normally.
- Turning encryption on also encrypts the bucket's existing objects in the background. ETag, Last-Modified and replication status are kept. The rate is `HONKIO_S3_ENCRYPTION_REWRITE_RATE_MB` (default 20 MiB/s), and progress is shown in the console's Encryption window. Blocks of the old plaintext file can stay on disk until the file system reuses them. If this is a problem, create a new encrypted bucket and copy the data into it.
- Turning encryption off applies only to new objects. Encrypted objects stay encrypted and readable, and the key is kept.

- Encryption is invisible to clients: size, ETag (MD5 of the plaintext data), Range and presigned URLs work as before. Responses include `x-amz-server-side-encryption: AES256`.
- Compression and encryption work together: data is first compressed and then encrypted. Multipart parts are encrypted already at upload time.
- **Not encrypted:** object names, metadata and tags. Data is sent to a replication target as plaintext (over TLS), so the target must encrypt it itself.
- **Loss of the Transit key = loss of the data.** Keep OpenBao backups in order. Do not delete the key `honkio-s3` and do not raise its `min_decryption_version` until the bucket keys have been re-encrypted (`transit/rewrap`).
- **Do not go back to a binary older than the version that added encryption** while encrypted objects exist.

7. All settings (environment variables)

Variable	Default	Description
<code>MINIO_ROOT_USER</code> , <code>MINIO_ROOT_PASSWORD</code>	(required)	Root credentials. Also <code>HONKIO_S3_ROOT_*</code> and the old <code>MINIO_ACCESS_KEY</code> / <code>MINIO_SECRET_KEY</code> . The user name needs at least 3 characters and the password at least 8.
<code>MINIO_ADDRESS</code>	<code>0.0.0.0:9000</code>	S3 port (also <code>--address</code> , <code>HONKIO_S3_ADDRESS</code>)
<code>HONKIO_S3_CONSOLE_ADDRESS</code>	off	Console port (also <code>--console-address</code>)
<code>MINIO_BROWSER</code>	on	<code>off</code> disables the console
<code>MINIO_BROWSER_REDIRECT_URL</code>	–	Public address of the console. The browser is redirected there from the S3 root.
<code>HONKIO_S3_TRUSTED_PROXIES</code>	–	Comma-separated CIDR networks whose <code>X-Forwarded-For</code> header is trusted. Private and loopback addresses are always trusted.
<code>MINIO_REGION</code>	<code>us-east-1</code>	Reported region (any region is accepted in signatures). Also <code>HONKIO_S3_REGION</code> , <code>MINIO_SITE_REGION</code> .
<code>MINIO_API_CORS_ALLOW_ORIGIN</code>	*	Allowed CORS origins, wildcards allowed (e.g. <code>https://*.example.com</code>)
<code>MINIO_PROMETHEUS_AUTH_TYPE</code>	token	<code>public</code> = metrics without a token
<code>HONKIO_S3_AUDIT_LOG</code>	off	<code>stdout</code> or a file path: one JSON line per request
<code>HONKIO_S3_PUBLIC_CACHE_CONTROL</code>	–	Cache-Control for anonymous read requests, if the object has none of its own
<code>HONKIO_S3_COMPRESSION</code>	off	<code>zstd</code> compresses new objects on disk (decompressed on the fly)
<code>HONKIO_S3_COMPRESSION_LEVEL</code>	1	<code>zstd</code> level –7 (fastest) ... 19
<code>HONKIO_S3_OPENBAO_ADDR</code>	off	OpenBao address. Enables bucket encryption.
<code>HONKIO_S3_OPENBAO_CACERT</code>	–	OpenBao CA certificate (PEM), if it is not publicly trusted
<code>HONKIO_S3_OPENBAO_TRANSIT_MOUNT</code>	<code>transit</code>	Path of the Transit engine
<code>HONKIO_S3_OPENBAO_TRANSIT_KEY</code>	<code>honkio-s3</code>	Transit key used to encrypt the keys of new buckets
<code>HONKIO_S3_OPENBAO_ROLE_ID_FILE</code> , <code>HONKIO_S3_OPENBAO_SECRET_ID_FILE</code>	–	Login with AppRole (path <code>HONKIO_S3_OPENBAO_APPROLE_MOUNT</code> , default <code>approle</code>)

Variable	Default	Description
<code>HONKIO_S3_OPENBAO_K8S_ROLE</code>	–	Login with Kubernetes auth (path <code>HONKIO_S3_OPENBAO_K8S_MOUNT</code> , default <code>kubernetes</code>)
<code>HONKIO_S3_OPENBAO_TOKEN_FILE</code>	–	Login with the OpenBao Agent token file (sink)
<code>HONKIO_S3_ENCRYPTION_REWRITE_RATE_MB</code>	20	Read rate for encrypting existing objects (MiB/s), 0 = unlimited
<code>HONKIO_S3_SCAN_INTERVAL_MINUTES</code>	15	Usage statistics and lifecycle
<code>HONKIO_S3_SCRUB_INTERVAL_DAYS</code>	30	Bitrot check interval, 0 = off
<code>HONKIO_S3_SCRUB_RATE_MB</code>	50	Bitrot check read rate (MiB/s)
<code>HONKIO_S3_STALE_UPLOAD_HOURS</code>	24	Removal of incomplete multipart uploads
<code>HONKIO_S3_REPLICATION_RESYNC_MINUTES</code>	360	Interval of full reconciliation, 0 = only at startup, after errors and on request
<code>HONKIO_S3_REPLICATION_RESYNC_RATE_MB</code>	20	Reconciliation copy rate (MiB/s), 0 = unlimited
<code>HONKIO_S3_DATA_DIR</code>	–	Data directory, if it is not given on the command line
<code>RUST_LOG</code>	<code>info</code>	Log level (e.g. <code>warn</code> , <code>debug</code> , <code>honkio_s3::replication=debug</code>)

For compatibility, Honkio S3 accepts the MinIO flags `--certs-dir`, `--quiet`, `--json` and `--anonymous`, but they do nothing. TLS is handled by a reverse proxy.

8. Supported S3 operations

Area	Support
Authentication	SigV4 headers, presigned URL (GET and PUT, <code>response-*</code> overrides), aws-chunked streaming (signed and unsigned, CRC32/CRC32C trailers), SigV2 for old clients. <code>Content-MD5</code> and <code>x-amz-content-sha256</code> are verified.
Buckets	List, Create, Head, Delete, GetBucketLocation, bucket policy, lifecycle, CORS and tagging (stored), default encryption SSE-S3 (<code>Get/Put/DeleteBucketEncryption</code> , AES256, keys in OpenBao). Versioning is always off.
Objects	Put, Get (Range, If-*), Head, Delete, DeleteObjects, CopyObject (COPY/REPLACE), tagging, ListObjects V1, V2 and Versions, conditional writes (<code>If-None-Match: *</code>)
Multipart	Create, UploadPart, UploadPartCopy, Complete, Abort, ListParts, ListMultipartUploads
MinIO admin	Users, policies and attaching them, quotas, <code>info</code> and import of service accounts
Not supported	Clustering and erasure coding, versioning, bucket notifications, per-object SSE header and SSE-C/SSE-KMS (encryption is set on the bucket), object lock, <code>mc replicate</code> , groups, STS, presigned POST, virtual-host addresses

4. Migrating from MinIO

Source: docs/en/MIGRATION.md

This guide describes how to move the objects, users and bucket settings of an existing MinIO to Honkio S3, and how to switch traffic to the new server in a controlled way that can be rolled back. The method has been used at Honkio (media41: about 65 buckets; MinIO and Honkio S3 side by side on the same machine).

The fastest and least painful way is `honkio-s3 migrate` (section 0): one command moves the users, the buckets with their settings, and the objects so that keys, ETags, timestamps and metadata are kept. You can run the command again, and it then copies only the changes.

Principles:

- **Keys are kept.** Users are imported from MinIO with the same access keys and secrets, so application settings are not touched. Presigned URLs created before the switch also keep working as long as the hostname stays the same.
- **Data is copied with the `honkio-s3 migrate` command** (or alternatively with `mc mirror`, sections 6–7). First everything is copied, and just before the switch a repeat run copies the changes.
- **The switch is made in the proxy.** Clients use the same address as before. The recommended way is to move **one bucket at a time** (section 9.1): a router in front of MinIO sends migrated buckets to Honkio S3 and the rest to MinIO, and each bucket is moved with its own write pause of a few seconds. A switch made all at once is described in section 9.2.
- **MinIO stays as a backup.** It is kept unchanged until Honkio S3 has been shown to work.

0. Quick migration: `honkio-s3 migrate` (recommended)

`honkio-s3 migrate` moves the whole MinIO with one command. You can run the command again, and it then copies only the changes made after the previous run. The migration is invisible to clients:

What moves	How
Users, service accounts, policies	From the MinIO IAM export with the same access keys and secrets. Application settings are not touched.
Groups	Honkio S3 has no groups, so the group's policies are attached directly to its members. Permissions stay the same.
Buckets	Same name and creation date
Bucket settings	Bucket policy (anonymous access, e.g. <code>public/*</code>), CORS, lifecycle, bucket tags, hard quota
Objects	Same key, ETag (including the multipart <code>-N</code> ETag) , Last-Modified to the millisecond , Content-Type, Cache-Control, Content-Disposition, Content-Encoding, Content-Language, Expires, <code>x-amz-meta-*</code> , object tags and the number of parts

Data is written directly to the Honkio S3 data directory, and that is why the ETag and timestamp are kept. An S3 copy (`mc mirror`) cannot keep them. The content of single-part objects is checked with MD5 against the source ETag. If disk compression is on (`HONKIO_S3_COMPRESSION=zstd`), objects are compressed already during the migration.

Features that Honkio S3 does not have are reported at the end of the run in a list "Not migrated / check by hand" (versioning, object lock, SSE encryption, MinIO replication rules, bucket notifications, a policy or lifecycle that was not migrated). They are not skipped silently. What they mean is described in section 2.

1. Install Honkio S3 following DEPLOYMENT.md, but do not send traffic to it yet. Set the same region as in MinIO (`MINIO_REGION`). The disk of the data directory must have room for the MinIO data plus 10 % headroom (less with compression on).

2. Credentials into the environment. The source credential is the MinIO root or an admin user, and it is given only in environment variables, never on the command line. Read the credentials from OpenBao so that they are not shown on screen:

```
sudo -i
export MIGRATE_SOURCE_ACCESS_KEY="$(bao kv get -field=user secret/minio/root)"
export MIGRATE_SOURCE_SECRET_KEY="$(bao kv get -field=password secret/minio/root)"
set -a; . /etc/honkio-s3/honkio-s3.env; . /etc/honkio-s3/secrets.env; set +a # Honkio S3 settings: root
user is not imported, compression
M="sudo --preserve-
env=MIGRATE_SOURCE_ACCESS_KEY,MIGRATE_SOURCE_SECRET_KEY,MINIO_ROOT_USER,HONKIO_S3_COMPRESSION,HONKIO_S3_CO
MPRESSION_LEVEL -u honkio-s3 honkio-s3 migrate --data /var/lib/honkio-s3 --source http://127.0.0.1:9000"
```

The command is run as the `honkio-s3` user so that the files get the right owner.

3. A dry run shows what would move and what must be checked by hand. It writes nothing:

```
$M --dry-run
```

4. Initial copy. Run it in `tmux`, because a large migration takes hours. `--rate-mb` limits the total speed so that MinIO production use does not slow down. Choose the value based on the disks (slow disk 5–10, SSD 50–100):

```
$M --rate-mb 20 2>&1 | tee /root/honkio-s3-migrate-$(date +%Y%m%d-%H%M).log
```

The output has a line for each bucket and a summary at the end. The exit code is 0 only if nothing failed. Failed objects are shown on lines `<bucket>/<key>: FAILED: <reason>`. A typical reason is a key that the file system does not allow: the same bucket has both `a` and `a/b`, or a part of the key is over 255 bytes. Rename them in MinIO and repeat the run.

The Honkio S3 service can be running during the migration. Its usage statistics are updated in the next scan (default 15 min).

5. Test the applications with a temporary hostname (section 8). Run `$M` again if needed. A repeat run skips identical objects (ETag, size and timestamp match) and copies only the changes.

6. Switch:

1. Write pause in the proxy (section 9.2, step 1). Reads work the whole time.
2. Final run with deletes: `$M --delete`. It copies only the changes, so it usually takes seconds or minutes, and it also brings in newly created users. `--delete` removes from Honkio S3 the objects that were deleted from MinIO after the initial copy.
3. Point the proxy at Honkio S3 and remove the write pause (section 9.2, step 3).
4. Smoke test (section 9.2, step 4). MinIO stays as a backup (section 10), and rollback is done as in section 11.

Other options:

Option	Explanation
<code>--bucket B</code>	Only this bucket (can be repeated). Also suits a one-bucket-at-a-time migration (section 9.1).
<code>--exclude-bucket B</code>	Skip a bucket (e.g. a service that is not migrated)
<code>--skip-iam</code>	Do not import users and policies
<code>--iam-dir DIR</code>	Use an unpacked <code>mc admin cluster iam export export</code> instead of the admin API (if the credential has no admin rights)
<code>--skip-objects</code>	Only users, buckets and their settings
<code>--workers N</code>	Parallel copies (default 4)
<code>--source-region R</code>	Region used to sign requests (default <code>us-east-1</code> ; a MinIO with its own region requires it)

The tool has been tested against a real MinIO (`tests/compat/migrate_test.py` , see README).

Sections 4–7 describe an older method based on `mc` commands. It is needed only for a one-bucket-at-a-time migration through the router (section 9.1), where MinIO and Honkio S3 serve side by side.

1. Phases and schedule

Phase	What is done	Duration	Downtime
1. Survey	Find out the MinIO contents and the features in use (section 2)	1 h	no
2. Honkio S3 installation	DEPLOYMENT.md, on its own port or its own machine	1–2 h	no
3. Users and policies	IAM export from MinIO and import into Honkio S3 (section 4)	15 min	no
4. Bucket settings	Anonymous access, quotas, lifecycle (section 5)	15 min	no
5. Initial copy	<code>mc mirror throttled</code> (section 6)	amount of data / speed, e.g. 500 GB at 20 MiB/s $\approx$ 7 h	no
6. Continuous sync	<code>honkio-s3-sync.service</code> (section 7)	for the duration of testing	no
7. Verification and testing	Comparison and application tests at the new address (section 8)	1 day –	no
8. Switch	One bucket at a time with the router (section 9.1) or all at once (section 9.2)	1–5 min per bucket / 5–30 min at once	only writes to the bucket in question (reads work)
9. Follow-up	MinIO to read-only mode as a backup, later shut down (section 10)	–	no

2. Survey: what is in MinIO

In the commands, `myminio` is the `mc` alias of MinIO and `honkio-s3` the `mc` alias of Honkio S3, both with root credentials. The aliases are created in section 3.

```
mc admin info myminio           # version, disks, status
mc du --depth 2 myminio         # data per bucket
mc ls myminio | wc -l           # number of buckets
mc admin user list myminio | wc -l # users
mc admin group list myminio     # groups (Honkio S3 does not support them, see below)
```

Check for features that Honkio S3 does not support or that move differently. The loop goes through every bucket and prints only the exceptions:

```
for b in $(mc ls --json myminio | sed -n 's/.*"key": "\([^"]*\)"\/".*\/1/p'); do
v=$(mc version info myminio/$b 2>/dev/null | grep -ciE 'enabled|suspended')
l=$(mc retention info --default myminio/$b 2>/dev/null | grep -ci 'mode')
e=$(mc encrypt info myminio/$b 2>/dev/null | grep -ci 'sse')
r=$(mc replicate ls myminio/$b 2>/dev/null | grep -ci 'arn')
n=$(mc event ls myminio/$b 2>/dev/null | grep -c 'arn')
i=$(mc ilm rule ls myminio/$b 2>/dev/null | grep -ciE 'expir|transition')
q=$(mc quota info --json myminio/$b 2>/dev/null | grep -c '"quota": [1-9]')
[ "$v$l$e$r$n$i$q" = "0000000" ] || echo "$b versioning=$v lock=$l encryption=$e replication=$r
events=$n ilm=$i quota=$q"
done
```

Finding in MinIO	What happens in Honkio S3	Action
Versioning on	Only the latest version of an object is copied. Honkio S3 has no versioning.	Accept, or leave the bucket in MinIO
Object lock / retention	The lock does not move	Find out whether the lock is a legal requirement. If it is, the bucket stays in MinIO.
SSE encryption	Objects are read decrypted and stored in Honkio S3 unencrypted	Encrypt the disk (LUKS) or leave the bucket in MinIO
Replication (mc replicate)	Does not move	Configure it again in the Honkio S3 console (Replication)
Bucket notifications	Not supported	Change the application to poll, or leave the bucket in MinIO
Lifecycle	Moves if the rules are of the Expiration, Prefix, Tag, ObjectSize or AbortIncompleteMultipartUpload type. Transition and version rules are rejected.	Section 5
Quota	Moves (hard quota)	Section 5
Groups	Not supported. Permissions granted through a group do not move.	Attach the policies directly to users before the export (section 4)
Virtual-host addresses (bucket.s3.example.com)	Not supported	Switch clients to path-style mode before the switch

3. Preparation

1. **Install Honkio S3** following DEPLOYMENT.md. - On the same machine it is installed on its own port (at Honkio 9300; MinIO stays on port 9000), on another machine on the default ports. - The disk of the data directory must have room for at least the MinIO data plus 10 % headroom. - Set the same region as in MinIO (MINIO_REGION). Honkio S3 accepts any region in signatures, but some clients read it.

2. **Root credential.** You can use the same root credential as in MinIO or a new one from OpenBao. Applications do not use the root credential.
3. **mc aliases** on the machine where the copy is run. The copy goes through **mc** (read and write), so it is most efficient to run it on the MinIO or Honkio S3 machine. The credentials are read from OpenBao, and they are not shown on screen:

```
sudo -i      # the aliases are stored in root's ~/.mc/config.json file (0600)
mc alias set myminio http://127.0.0.1:9000 "$(bao kv get -field=user secret/minio/root)" "$(bao kv get -
field=password secret/minio/root)"
mc alias set honkio-s3 http://127.0.0.1:9300 "$(bao kv get -field=user secret/honkio-s3/root)" "$(bao kv
get -field=password secret/honkio-s3/root)"
mc admin info honkio-s3
```

Use **mc** version 2023 or newer. Older versions lack the command **mc admin cluster iam export**.

4. Users and policies

The MinIO IAM export contains the users **with their secrets**, policies, policy attachments and service accounts. Honkio S3 imports them as they are, so the keys stay the same.

```
# 1) Groups: if `mc admin group list myminio` is not empty, attach the group's policies directly
# to its members, because Honkio S3 does not support groups:
# mc admin group info myminio <group> → policies and members
# mc admin policy attach myminio <policy> --user <member>

# 2) Export (the zip contains all secrets: keep it readable by root only)
install -d -m 0700 /root/honkio-s3-migration && cd /root/honkio-s3-migration
mc admin cluster iam export myminio          # → myminio-iam-info.zip
unzip -o myminio-iam-info.zip                 # → iam-assets/users.json, policies.json, ...

# 3) Import. Honkio S3 should be stopped, because the command writes the IAM file directly.
systemctl stop honkio-s3
sudo -u honkio-s3 honkio-s3 admin --data /var/lib/honkio-s3 import-minio-iam /root/honkio-s3-
migration/iam-assets
systemctl start honkio-s3
# → "imported N users, M service accounts, K policies"

# 4) Check
mc admin user list honkio-s3 | wc -l          # same number as in MinIO
mc admin policy list honkio-s3
```

The import merges the export with the existing Honkio S3 users: users with the same name are overwritten, and the others are kept. The root user is not imported. So you can run the import again as many times as needed.

If the **honkio-s3** user cannot read the **/root** directory, copy **iam-assets** to a directory it can read, and delete the copy right after the import. Alternatively, run the import as root and fix the ownership afterwards: **chown -R honkio-s3:honkio-s3 /var/lib/honkio-s3/.honkio-s3.sys/iam**.

Important: New users created in MinIO during the sync period do not move with **mc mirror**. For example, **honkio-server** creates a user for a new application automatically. **Repeat the export and import just before the switch** (section 9.2, step 2). When moving one bucket at a time, **honkio-s3-iam-sync.timer** does this automatically (section 9.1).

5. Bucket settings

Buckets and objects come with `mc mirror` (section 6). The buckets' own settings are copied separately. Run this after the initial copy, when the buckets exist, and again before the switch:

```
for b in $(mc ls --json myminio | sed -n 's/.*"key":\[^\]*\\\/".*\/\1/p'); do
  mc mb --ignore-existing honkio-s3/$b >/dev/null
  # Anonymous access (e.g. public/* read access); "{}" = no policy
  mc anonymous get-json myminio/$b > /tmp/anon-$b.json 2>/dev/null
  if grep -q '"Effect"' /tmp/anon-$b.json; then mc anonymous set-json /tmp/anon-$b.json honkio-s3/$b; fi
  rm -f /tmp/anon-$b.json
  # Quota (in bytes; 0 or missing = no quota)
  q=$(mc quota info --json myminio/$b 2>/dev/null | sed -n 's/.*"quota":\[^\]*\\\/".*\/\1/p')
  [ "${q:-0}" -gt 0 ] && mc quota set honkio-s3/$b --size "$q"
  # Lifecycle
  if mc ilm rule ls myminio/$b >/dev/null 2>&1; then
    mc ilm rule export myminio/$b > /tmp/ilm-$b.json && mc ilm rule import honkio-s3/$b <
    /tmp/ilm-$b.json; rm -f /tmp/ilm-$b.json
  fi
done
```

Finally check: `mc anonymous list honkio-s3/<bucket>` and the Access and Size/quota columns of the bucket list in the console.

6. Initial copy

```
# Throttled copy: copying the data must not slow down production.
nice -n 19 ionice -c3 mc mirror --overwrite --preserve --json --limit-upload 20MiB myminio honkio-s3 \
  > /root/honkio-s3-migration/mirror-$(date +%Y%m%d-%H%M).jsonl 2>&1
```

- `--limit-upload` limits the write speed. Choose the value based on the disks: on Honkio's 1 vCPU machine with a slow disk, an unthrottled copy raised MinIO response times to about half a second, and 5 MiB/s was safe. Watch the response times (`mc admin trace`, metrics) and adjust the speed.
- Run the copy in a `tmux` or `screen` session or as a `systemd` service. It takes hours.
- You can stop the copy and start it again: `--overwrite` copies only missing and changed objects.
- An interrupted run can leave incomplete multipart uploads in Honkio S3. They are cleaned up automatically after one day (`HONKIO_S3_STALE_UPLOAD_HOURS`). They show in disk usage but not in listings.
- Errors show in the log: `grep '"status":"error"' /root/honkio-s3-migration/mirror-*.jsonl`. Typical reasons:
- A key that the file system does not allow: a part over 255 bytes, or `__XLDIR__` as part of the key.
- The same bucket has both `a` and `a/b`. Both cannot exist at the same time (`XMinioParentIsObject`).
- The quota is full, if quotas were set before the copy.

Sort out such objects with the application (rename them) before the switch.

7. Continuous sync

After the initial copy, a sync service keeps Honkio S3 up to date with MinIO. MinIO changes (uploads, overwrites and deletes) are copied within seconds. The service is ready in the file `deploy/migration/honkio-s3-sync.service`.

```
install -m 0644 deploy/migration/honkio-s3-sync.service /etc/systemd/system/
systemctl daemon-reload && systemctl enable --now honkio-s3-sync
journalctl -u honkio-s3-sync -f # a JSON line for each copied or deleted object
```

- `--remove` deletes from Honkio S3 the objects that were deleted from MinIO, and **also whole buckets that exist only in Honkio S3** (observed: a bucket created only in Honkio S3 was emptied and deleted in 6 seconds). So nothing of your own may be written to Honkio S3 during the sync. If Honkio S3 needs its own bucket before the migration ends (e.g. for backups), add its name to the setting `SYNC_EXCLUDE_BUCKETS` in the file `/etc/honkio-s3/migration.conf` and restart `honkio-s3-sync` **before** creating the bucket.
- A restart of the service goes through the whole listing at the start and copies the differences. So a restart is safe.

8. Verification and testing

Comparison. The script `deploy/migration/verify-migration.sh` compares the object count, total size and `mc diff` result of each bucket. It changes nothing.

```
sh deploy/migration/verify-migration.sh myminio honkio-s3      # all buckets
sh deploy/migration/verify-migration.sh myminio honkio-s3 images # only the given ones
```

BUCKET	SRC_OBJECTS	DST_OBJECTS	SRC_BYTES	DST_BYTES
DIFFS				
65e847efbbef28198593a39c0	69	69	9645105	9645105
66210280ab3c9719bb3b7bf00	200	200	81527373	81527373
OK: all buckets identical				

While the sync is running, objects that just changed can show briefly as differences. Run the comparison again or do it during the write pause (section 9).

Content spot check. The object contents are identical: `mc cat myminio/b/k | sha256sum` and `mc cat honkio-s3/b/k | sha256sum`.

Application tests at the new address. Open a temporary hostname for Honkio S3, for example `s3-new.example.com` (at Honkio `media-new.honkio.com`). With it, test using the applications' credentials:

- upload, download, listing and delete in a test bucket
- presigned GET and PUT
- anonymous read from a public path, for example `curl https://s3-new.example.com/<bucket>/public/<image>`, and the 403 of a private path
- the application's own automatic bucket and user creation, if the application uses the MinIO admin API.

9. Switch

9.1 One bucket at a time with the router (recommended)

The tool `deploy/migration/honkio-s3-cutover` moves buckets one at a time. Choose a small, low-risk bucket first, and continue the migration once it has been shown to work. Each bucket can be returned to MinIO with one command.

Structure:

```
clients → (edge proxy) → :9000 —firewalld forward-port→ nginx router :9080
                                     |— migrated buckets → Honkio S3 :9300
—replication→ MinIO
```

- **The router** (`nginx-honkio-s3-router.conf`, port 9080) sends the path `/<bucket>` and `/<bucket>/...` to Honkio S3 if the bucket has been migrated, and everything else to MinIO. The Host header, the `X-Forwarded-*` headers and the server's `Server` header are passed unchanged. So signatures and presigned URLs work in both directions.
- **Traffic steering:** the clients' address does not change. `firewalld` sends traffic arriving from outside on port 9000 to the router (`forward-port 9000 → 9080`). Local traffic on the machine to `127.0.0.1:9000`, such as `mc` and the sync, still goes directly to MinIO. MinIO does not need a restart, and the edge proxy settings do not need to change.
- **Sync MinIO → Honkio S3** (`honkio-s3-sync.service = honkio-s3-cutover sync-run`) skips migrated buckets (`--exclude-bucket`). Tested: nothing is copied to or deleted from a skipped bucket.
- **Reverse replication Honkio S3 → MinIO:** during the migration, the same bucket in MinIO is added as the Honkio S3 replication target for the bucket. The MinIO copy stays up to date, so returning the bucket is only a routing switch. The first reconciliation only records the identical objects that already exist and does not copy them again.
- **Users and policies** are managed in MinIO during the migration. The admin API `/minio/admin` goes to MinIO, so for example `honkio-server` creates new applications there as before. `honkio-s3-iam-sync.timer` copies them to Honkio S3 every 10 minutes. It does not delete users from Honkio S3 that were deleted from MinIO.

Installation on the migration machine (as root; needs `nginx`, `firewalld`, the `mc` aliases `myminio` and `honkio-s3`, and Python 3):

```
cd <repo>/deploy/migration
./test-honkio-s3-cutover.sh           # self-test on separate ports (19000-19080); everything must pass
install -m 0644 /dev/null /etc/honkio-s3/migration.conf # and fill in the settings (see the start of the
honkio-s3-cutover file)
./honkio-s3-cutover install           # router :9080 (no traffic yet), units, /usr/local/sbin/honkio-s3-
cutover
systemctl restart honkio-s3-sync      # new sync unit
honkio-s3-cutover iam-sync            # users and policies right away
```

Before enabling the router, check that the router returns the same response as a direct request:

```
curl -s -D - -o a -H "Host: media.example.com" http://127.0.0.1:9000/<bucket>/<public-object>
curl -s -D - -o b -H "Host: media.example.com" http://127.0.0.1:9080/<bucket>/<public-object>
cmp a b
```

Enabling the router: `honkio-s3-cutover router-on`. Everything still goes to MinIO, but through the router. Watch the logs (`/var/log/nginx/honkio-s3-router.access.log` and `.error.log`). You can disable the router with the command `honkio-s3-cutover router-off`, but only when no bucket has been migrated.

Migrating a bucket: `honkio-s3-cutover cutover <bucket> [<bucket> ...]`. For each bucket the tool:

1. copies the IAM from MinIO
2. stops writes to the bucket in the router (503, reads work)
3. restarts the sync without the bucket
4. runs a final `mc mirror` and copies the bucket settings (anonymous access, quota, lifecycle)
5. compares the bucket on both sides. If the copies differ, the bucket stays on MinIO, and the tool stops with an error.
6. adds reverse replication to MinIO and waits for its first reconciliation

- switches the bucket to Honkio S3 and opens writes
- checks that the bucket responds with `Server: Honkio S3`.

Rollback: `honkio-s3-cutover rollback <bucket>` stops writes and updates MinIO from Honkio S3 (`mc mirror --remove`). After that it compares the copies, removes the reverse replication, switches the bucket to MinIO and takes it back into the sync.

Status: `honkio-s3-cutover status` shows the state of the router, the sync and the timer, plus the migrated buckets and the server that responds for them. The tool's log is in the file `/var/log/honkio-s3-cutover.log`.

When all buckets have been migrated: only the admin API (`/minio/admin`, e.g. `honkio-server` user creation) and the bucket listing still go to MinIO.

- Run `honkio-s3-cutover iam-sync` one last time.
- Change the router's default route (`location / → proxy_pass http://honkio_s3_local;`) and run `nginx -t && systemctl reload nginx`. From now on users are managed in Honkio S3.
- Stop and disable `honkio-s3-sync` and `honkio-s3-iam-sync.timer`.
- Reverse replication toward MinIO can stay on until MinIO is removed (section 10).

Later you can point the edge proxy directly at Honkio S3 and remove the router (`router-off --force`).

9.2 All at once

The switch is made during a short **write pause**. Reads work the whole time, and writes return 503 for a moment. Client libraries retry on 503.

1. Write pause in the proxy (nginx, in the MinIO server block):

```
location / {
    if ($request_method !~ ^(GET|HEAD|OPTIONS)$) { return 503; }
    proxy_pass http://127.0.0.1:9000;      # still MinIO
    # ... other settings unchanged
}
```

```
nginx -t && systemctl reload nginx
```

2. Final syncs:

```
# New users and policies (section 4)
mc admin cluster iam export myminio && unzip -o myminio-iam-info.zip
systemctl stop honkio-s3 && sudo -u honkio-s3 honkio-s3 admin --data /var/lib/honkio-s3 import-minio-iam
iam-assets && systemctl start honkio-s3
# Bucket settings (section 5, same loop)
# Data: wait until the sync has processed everything, then stop it
systemctl stop honkio-s3-sync
mc mirror --overwrite --remove --preserve myminio honkio-s3      # safety check, usually copies nothing
sh deploy/migration/verify-migration.sh myminio honkio-s3      # → OK: all buckets identical
```

3. Point the proxy at Honkio S3 and remove the write pause:

```
location / {
    proxy_pass http://127.0.0.1:9300;      # Honkio S3
    # ... other settings unchanged; write block removed
}
```

```
nginx -t && systemctl reload nginx
curl -sI https://s3.example.com/ | grep -i '^server'      # Server: Honkio S3
```

If you use a DNS change instead of the proxy, lower the TTL (e.g. 60 s) already a day before the switch. Remember that because of DNS caches, some clients still write to the old address for a while. That is why the write pause should stay in effect in MinIO until the old TTL has passed.

4. Smoke test right after the switch: application upload and download, a public image, `journalctl -u honkio-s3` (no errors) and the Monitoring view of the console.

If the same MinIO is also used by a service that is not to be migrated yet (e.g. GitLab), its buckets can be left in MinIO with a path-based proxy rule (`location /gitlab-` → MinIO).

10. After the switch

- **MinIO to read-only mode as a backup:** keep MinIO running, but do not write to it. Keep it for example for 2–4 weeks.
- **Monitoring:** health, metrics and alerts (DEPLOYMENT.md, section 7).
- **Backup or replication** for Honkio S3, and test the restore.
- **Decommissioning MinIO:** when Honkio S3 has worked for the agreed time, shut down MinIO, take a final backup of its data directory and remove the service. Also delete `/root/honkio-s3-migration` (it contains the IAM export with secrets).

11. Rollback

Situation	Rollback
The switch has not been made yet	No effect on clients. Stop <code>honkio-s3-sync</code> , and Honkio S3 can be emptied and started over.
The switch has been made, and nothing has been written to Honkio S3 yet	Switch the proxy back to MinIO (<code>proxy_pass</code> → 9000) and reload nginx.
Data has already been written to Honkio S3	1) Write pause (section 9.2, step 1). 2) Copy the changes back: <code>mc mirror --overwrite --preserve honkio-s3 myminio</code> (without <code>--remove</code> , unless you also want the Honkio S3 deletes in MinIO). 3) Proxy to MinIO. 4) New users created in Honkio S3 are created in MinIO by hand.

12. Differences clients may notice

- **honkio-s3 migrate:** ETag and Last-Modified are kept for all objects. If the data has already been copied with `mc mirror`, `honkio-s3 migrate` fixes the timestamps and metadata afterwards without copying the data again. Multipart objects whose ETag changed are copied again.
- **mc mirror:** Last-Modified is the copy time and not the original time, because `mc` writes the objects again (checked on media41: ETag and Content-Type were kept, the timestamp changed). For large multipart objects the ETag can change, because `mc` splits the parts into different sizes. The content is still identical.
- **Versions, locks and encryption** do not move (section 2).
- **Region** is the one set with the `MINIO_REGION` variable.
- **Error messages and headers** are S3-compatible. The server name is `Server: Honkio S3`.

13. Checklist

- ☐ Survey done (`honkio-s3 migrate --dry-run` or section 2), and there are no unsupported features (or a decision has been made about them).
- ☐ Honkio S3 installed, monitored, and there is enough disk space.
- ☐ IAM imported, and the number of users matches. Group permissions have been attached directly to users.
- ☐ Bucket settings (anonymous access, quotas, lifecycle) copied.
- ☐ Initial copy done, and the log has no errors or they have been sorted out.
- ☐ `honkio-s3-sync` running, and `verify-migration.sh` shows OK.
- ☐ Applications tested with the temporary hostname.
- ☐ Switch window agreed, the nginx change for the write pause ready and the rollback written down.
- ☐ After the switch: smoke test, MinIO in read-only mode, backup or replication on.
- ☐ Cleanup: `honkio-s3-sync` disabled, `/root/honkio-s3-migration` deleted and MinIO shut down as agreed.

Appendix A: systemd service

File in the repository: `deploy/systemd/honkio-s3.service`

```
[Unit]
Description=Honkio S3 Server (S3-compatible object storage)
Documentation=https://gitlab.honkio.net/honkio-oss/honkio-s3/-/blob/main/docs/DEPLOYMENT.md
After=network-online.target openbao-agent-honkio-s3.service
Wants=network-online.target
# With OpenBao: wait for the agent that renders /etc/honkio-s3/secrets.env.
# Without OpenBao this line is harmless (the unit simply does not exist).
Wants=openbao-agent-honkio-s3.service

[Service]
User=honkio-s3
Group=honkio-s3
# Plain settings (ports, paths, tuning) and secrets in separate files:
# secrets.env is written by the OpenBao agent (or by hand, mode 0600 root).
EnvironmentFile=/etc/honkio-s3/honkio-s3.env
EnvironmentFile=/etc/honkio-s3/secrets.env
ExecStart=/usr/local/bin/honkio-s3 server /var/lib/honkio-s3
Restart=always
RestartSec=3
LimitNOFILE=65536
TimeoutStopSec=60
MemoryMax=1G

# Hardening: the server only needs its data directory.
NoNewPrivileges=true
ProtectSystem=strict
ProtectHome=true
PrivateTmp=true
PrivateDevices=true
ProtectKernelTunables=true
ProtectKernelModules=true
ProtectControlGroups=true
RestrictSUIDSGID=true
LockPersonality=true
ReadWritePaths=/var/lib/honkio-s3
# Uncomment when HONKIO_S3_AUDIT_LOG points to a file under /var/log/honkio-s3:
#ReadWritePaths=/var/log/honkio-s3

[Install]
WantedBy=multi-user.target
```

Appendix B: Settings file (no secrets)

File in the repository: `deploy/systemd/honkio-s3.env.example`

```
# /etc/honkio-s3/honkio-s3.env: Honkio S3 settings WITHOUT secrets (owner root, mode 0644).
# Secrets (MINIO_ROOT_USER, MINIO_ROOT_PASSWORD) go to /etc/honkio-s3/secrets.env,
# written by the OpenBao agent: see docs/DEPLOYMENT.md.

# S3 API. Bind to 127.0.0.1 when the reverse proxy runs on the same host;
# 0.0.0.0 when the proxy is elsewhere (then limit the port with the firewall).
MINIO_ADDRESS=127.0.0.1:9000

# Web console on its own port and its own hostname (never the S3 hostname).
# Leave unset (or MINIO_BROWSER=off) to run without a console.
HONKIO_S3_CONSOLE_ADDRESS=127.0.0.1:9001
# Public console URL: browsers opening the S3 root are redirected there.
MINIO_BROWSER_REDIRECT_URL=https://s3-console.example.com

# Proxies whose X-Forwarded-For is trusted (private/loopback ranges always are).
# Needed only when the proxy connects from a public / non-RFC1918 address.
#HONKIO_S3_TRUSTED_PROXIES=203.0.113.0/24

# Region reported to clients (any region is accepted in signatures).
MINIO_REGION=us-east-1

# CORS for browser uploads: * or a comma list, wildcards allowed.
MINIO_API_CORS_ALLOW_ORIGIN=*

# Access log: one JSON line per request to journald (stdout) or a file.
HONKIO_S3_AUDIT_LOG=stdout

# Cache-Control for anonymous reads of objects that have none.
#HONKIO_S3_PUBLIC_CACHE_CONTROL=public, max-age=86400

# Compress new objects on disk with zstd (decompressed on the fly when read).
# Already-compressed formats (images, video, zip) are stored as-is.
#HONKIO_S3_COMPRESSION=zstd
#HONKIO_S3_COMPRESSION_LEVEL=1
# Bucket encryption: keys wrapped by OpenBao Transit (see docs/DEPLOYMENT.md).
#HONKIO_S3_OPENBAO_ADDR=https://openbao.example.com:8200
#HONKIO_S3_OPENBAO_ROLE_ID_FILE=/etc/honkio-s3/openbao/role_id
#HONKIO_S3_OPENBAO_SECRET_ID_FILE=/etc/honkio-s3/openbao/secret_id

# Prometheus: without this, scraping needs a token from `mc admin prometheus generate`.
#MINIO_PROMETHEUS_AUTH_TYPE=public

# Background jobs (defaults shown). Lower the rates on slow disks.
#HONKIO_S3_SCAN_INTERVAL_MINUTES=15
#HONKIO_S3_SCRUB_INTERVAL_DAYS=30
#HONKIO_S3_SCRUB_RATE_MB=50
#HONKIO_S3_STALE_UPLOAD_HOURS=24
#HONKIO_S3_REPLICATION_RESYNC_MINUTES=360
#HONKIO_S3_REPLICATION_RESYNC_RATE_MB=20

RUST_LOG=info
```

Appendix C: OpenBao policy: reading secrets

File in the repository: `deploy/openbao/honkio-s3-read.hcl`

```
# OpenBao policy: read-only access to Honkio S3 secrets (KV v2 mounted at "secret").
# bao policy write honkio-s3-read deploy/openbao/honkio-s3-read.hcl
path "secret/data/honkio-s3/*" {
  capabilities = ["read"]
}
path "secret/metadata/honkio-s3/*" {
  capabilities = ["read", "list"]
}
```

Appendix D: OpenBao policy: bucket encryption keys

File in the repository: `deploy/openbao/honkio-s3-transit.hcl`

```
# OpenBao policy: Honkio S3 may wrap and unwrap the data keys of encrypted
# buckets with the Transit key "honkio-s3", and nothing else. The Transit key
# itself never leaves OpenBao.
# bao policy write honkio-s3-transit deploy/openbao/honkio-s3-transit.hcl
path "transit/encrypt/honkio-s3" {
  capabilities = ["update"]
}
path "transit/decrypt/honkio-s3" {
  capabilities = ["update"]
}
```

Appendix E: OpenBao Agent configuration

File in the repository: `deploy/openbao/agent.hcl`

```
# OpenBao agent for Honkio S3 on a VM: Logs in with AppRole and renders the root
# credentials into /etc/honkio-s3/secrets.env. When the secret changes in
# OpenBao, the file is rewritten and Honkio S3 restarted.
#
#   bao agent -config=/etc/openbao-agent/honkio-s3/agent.hcl
# (run by deploy/openbao/openbao-agent-honkio-s3.service)

vault {
  # Your OpenBao address. For a private CA add: ca_cert = "/etc/openbao-agent/honkio-s3/ca.pem"
  address = "https://openbao.example.com:8200"
  retry {
    num_retries = 5
  }
}

auto_auth {
  method "approle" {
    config = {
      role_id_file_path      = "/etc/openbao-agent/honkio-s3/role_id"
      secret_id_file_path    = "/etc/openbao-agent/honkio-s3/secret_id"
      remove_secret_id_file_after_reading = false
    }
  }
}

template_config {
  # Re-read static (KV) secrets this often; a change re-renders the file.
  static_secret_render_interval = "5m"
  exit_on_retry_failure         = true
}

template {
  source      = "/etc/openbao-agent/honkio-s3/secrets.env.ctmpl"
  destination = "/etc/honkio-s3/secrets.env"
  perms       = "0600"
  # Honkio S3 reads its credentials at startup: restart it when they change.
  exec {
    command = ["systemctl", "try-restart", "honkio-s3.service"]
    timeout = "60s"
  }
}
```

Appendix F: OpenBao Agent template for the secrets file

File in the repository: `deploy/openbao/secrets.env.ctmpl`

```
{{- /* Rendered by the OpenBao agent to /etc/honkio-s3/secrets.env (mode 0600, root). */ -}}
{{- with secret "secret/data/honkio-s3/root" }}
MINIO_ROOT_USER={{ .Data.data.user }}
MINIO_ROOT_PASSWORD={{ .Data.data.password }}
{{- end }}
```

Appendix G: OpenBao Agent service

File in the repository: `deploy/openbao/openbao-agent-honkio-s3.service`

```
[Unit]
Description=OpenBao agent for Honkio S3 secrets
After=network-online.target
Wants=network-online.target
Before=honkio-s3.service

[Service]
# Runs as root: it writes the root-only secrets file and restarts Honkio S3.
ExecStart=/usr/bin/bao agent -config=/etc/openbao-agent/honkio-s3/agent.hcl
Restart=on-failure
RestartSec=5
NoNewPrivileges=true
ProtectHome=true
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

Appendix H: Kubernetes and External Secrets Operator

File in the repository: `k8s/honkio-s3-openbao.example.yaml`

```
# Honkio S3 in Kubernetes with the root credentials from OpenBao via the
# External Secrets Operator (ESO). Replace: namespace, OpenBao address,
# image, storage class/size, hostnames and the VPN ranges.
# Setup of the OpenBao side (policy, Kubernetes auth role): docs/DEPLOYMENT.md, section 6.
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eso-honkio-s3
  namespace: storage
---
apiVersion: external-secrets.io/v1
kind: SecretStore
metadata:
  name: openbao-honkio-s3
  namespace: storage
spec:
  provider:
    vault:
      server: https://openbao.example.com:8200 # or the in-cluster service, e.g. http://openbao-
active.openbao.svc:8200
      path: secret
      version: v2
      auth:
        kubernetes:
          mountPath: kubernetes
          role: eso-honkio-s3
          serviceAccountRef:
            name: eso-honkio-s3
---
apiVersion: external-secrets.io/v1
kind: ExternalSecret
metadata:
  name: honkio-s3-root
  namespace: storage
spec:
  refreshInterval: 15m
  secretStoreRef:
    kind: SecretStore
    name: openbao-honkio-s3
  target:
    name: honkio-s3-root
    creationPolicy: Owner
  data:
    - secretKey: MINIO_ROOT_USER
      remoteRef:
        key: honkio-s3/root
        property: user
    - secretKey: MINIO_ROOT_PASSWORD
      remoteRef:
        key: honkio-s3/root
        property: password
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: honkio-s3-config
  namespace: storage
data:
  MINIO_ADDRESS: "0.0.0.0:9000"
  HONKIO_S3_CONSOLE_ADDRESS: "0.0.0.0:9001"
```

```

MINIO_BROWSER_REDIRECT_URL: "https://s3-console.example.com"
MINIO_REGION: "us-east-1"
HONKIO_S3_AUDIT_LOG: "stdout"
RUST_LOG: "info"
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: honkio-s3
  namespace: storage
spec:
  serviceName: honkio-s3
  replicas: 1 # exactly one: the data directory has a single writer
  selector:
    matchLabels: { app: honkio-s3 }
  template:
    metadata:
      labels: { app: honkio-s3 }
    spec:
      securityContext:
        runAsUser: 1000
        runAsGroup: 1000
        fsGroup: 1000
      terminationGracePeriodSeconds: 60
      containers:
        - name: honkio-s3
          image: registry.example.com/honkio-s3:965b600 # built from this repo: docker build -t ...
          args: ["server", "/data"]
          envFrom:
            - configMapRef: { name: honkio-s3-config }
            - secretRef: { name: honkio-s3-root }
          ports:
            - { name: s3, containerPort: 9000 }
            - { name: console, containerPort: 9001 }
          livenessProbe:
            httpGet: { path: /minio/health/live, port: s3 }
            periodSeconds: 20
          readinessProbe:
            httpGet: { path: /minio/health/ready, port: s3 }
            periodSeconds: 10
          resources:
            requests: { cpu: 100m, memory: 128Mi }
            limits: { memory: 1Gi }
          volumeMounts:
            - { name: data, mountPath: /data }
      volumeClaimTemplates:
        - metadata:
            name: data
          spec:
            accessModes: ["ReadWriteOnce"]
            resources:
              requests:
                storage: 100Gi
---
apiVersion: v1
kind: Service
metadata:
  name: honkio-s3
  namespace: storage
spec:
  selector: { app: honkio-s3 }
  ports:
    - { name: s3, port: 9000, targetPort: s3 }
    - { name: console, port: 9001, targetPort: console }
---
# S3 API: public (or internal) hostname. Uploads are streamed, no size limit.
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: honkio-s3-s3

```

```

namespace: storage
annotations:
  cert-manager.io/cluster-issuer: letsencrypt-issuer
  nginx.ingress.kubernetes.io/proxy-body-size: "0"
  nginx.ingress.kubernetes.io/proxy-request-buffering: "off"
  nginx.ingress.kubernetes.io/proxy-buffering: "off"
  nginx.ingress.kubernetes.io/proxy-read-timeout: "3600"
  nginx.ingress.kubernetes.io/proxy-send-timeout: "3600"
spec:
  ingressClassName: nginx
  tls:
    - hosts: ["s3.example.com"]
      secretName: honkio-s3-s3-tls
  rules:
    - host: s3.example.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service: { name: honkio-s3, port: { name: s3 } }
---
# Web console: its own hostname, Limited to admin/VPN networks.
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: honkio-s3-console
  namespace: storage
  annotations:
    cert-manager.io/cluster-issuer: letsencrypt-issuer
    nginx.ingress.kubernetes.io/whitelist-source-range: "10.0.0.0/8"
    nginx.ingress.kubernetes.io/proxy-body-size: "0"
    nginx.ingress.kubernetes.io/proxy-request-buffering: "off"
    nginx.ingress.kubernetes.io/proxy-read-timeout: "3600"
    nginx.ingress.kubernetes.io/proxy-send-timeout: "3600"
spec:
  ingressClassName: nginx
  tls:
    - hosts: ["s3-console.example.com"]
      secretName: honkio-s3-console-tls
  rules:
    - host: s3-console.example.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service: { name: honkio-s3, port: { name: console } }

```

Appendix I: Dockerfile

File in the repository: Dockerfile

```
# Static musl build -> minimal runtime image.
FROM rust:1.99-alpine AS builder
RUN apk add --no-cache musl-dev
WORKDIR /src
COPY Cargo.toml Cargo.lock ./
COPY src ./src
RUN cargo build --release --locked && strip target/release/honkio-s3

FROM alpine:3.22
RUN apk add --no-cache ca-certificates && adduser -D -u 1000 honkio-s3 && mkdir /data && chown honkio-s3 /data
COPY --from=builder /src/target/release/honkio-s3 /usr/local/bin/honkio-s3
USER honkio-s3
# 9000 = S3 API, 9001 = web console (when HONKIO_S3_CONSOLE_ADDRESS is set)
EXPOSE 9000 9001
VOLUME ["/data"]
HEALTHCHECK --interval=30s --timeout=3s CMD wget -qO- http://127.0.0.1:9000/minio/health/live || exit 1
ENTRYPOINT ["honkio-s3"]
CMD ["server", "/data", "--address", ":9000"]
```

Appendix J: Migration tool honkio-s3-cutover

File in the repository: `deploy/migration/honkio-s3-cutover`

```
#!/bin/bash
# honkio-s3-cutover: gradual MinIO -> Honkio S3 migration, one bucket at a time.
# Guide: docs/MIGRAATIO.md, chapter "Siirto bucket kerrallaan".
#
# How it works
# - An nginx router (port 9080) sits in front of MinIO: external traffic to
#   the MinIO port is redirected to it with a firewall forward-port, so
#   clients keep their endpoint. Local traffic to 127.0.0.1:9000 (mc, the
#   sync) is not redirected.
# - Every bucket goes to MinIO until `cutover` routes it to Honkio S3.
# - The MinIO -> Honkio S3 sync (honkio-s3-sync.service) skips moved buckets.
# - Honkio S3 replicates each moved bucket back to MinIO, so `rollback` is a
#   routing change too.
# - IAM (users, policies) stays managed in MinIO during the migration and is
#   copied to Honkio S3 by `iam-sync` (timer: honkio-s3-iam-sync.timer).
#
# Commands (run as root)
#   install          install router, units and this tool from the repo directory
#   status           router, sync, timers and the state of moved buckets
#   router-on | router-off start / stop sending external traffic through the router
#   cutover <bucket>... move buckets to Honkio S3 (short write freeze per bucket)
#   rollback <bucket>... move buckets back to MinIO
#   iam-sync         copy users and policies from MinIO to Honkio S3
#   verify [bucket...] compare buckets (objects, size, mc diff)
#   sync-run        (used by honkio-s3-sync.service) mirror MinIO -> Honkio S3 except moved
#                   buckets
set -euo pipefail

CONF=${HONKIO_S3_CUTOVER_CONF:-/etc/honkio-s3/migration.conf}
# shellcheck disable=SC1090
[ -f "$CONF" ] && . "$CONF"
SRC_ALIAS=${SRC_ALIAS:-myminio} # mc alias of MinIO (root)
DST_ALIAS=${DST_ALIAS:-honkio-s3} # mc alias of Honkio S3 (root)
MINIO_URL=${MINIO_URL:-http://127.0.0.1:9000} # MinIO as seen from Honkio S3 (replication target)
HONKIO_S3_URL=${HONKIO_S3_URL:-http://127.0.0.1:9300}
CONSOLE_URL=${CONSOLE_URL:-http://127.0.0.1:9301}
HONKIO_S3_ENV=${HONKIO_S3_ENV:-/etc/default/honkio-s3} # Honkio S3 root credentials
MINIO_ENV=${MINIO_ENV:-/etc/default/minio} # MinIO root credentials
HONKIO_S3_DATA=${HONKIO_S3_DATA:-/data/honkio-s3}
HONKIO_S3_USER=${HONKIO_S3_USER:-honkio-s3}
PUBLIC_PORT=${PUBLIC_PORT:-9000}
ROUTER_PORT=${ROUTER_PORT:-9080}
FIREWALL_ZONE=${FIREWALL_ZONE:-public}
HOST_HEADER=${HOST_HEADER:-media.honkio.com} # public hostname, for smoke tests
SYNC_LIMIT=${SYNC_LIMIT:-5MiB}
SYNC_EXCLUDE_BUCKETS=${SYNC_EXCLUDE_BUCKETS:-} # Honkio S3-only buckets the sync must not
touch (space separated)
SYNC_WATCH=${SYNC_WATCH:-yes} # no = one full pass per run (source without change
notifications)
SYNC_UNIT=${SYNC_UNIT:-honkio-s3-sync.service}
REVERSE_REPLICATION=${REVERSE_REPLICATION:-yes}
CUTOVER_IAM_SYNC=${CUTOVER_IAM_SYNC:-yes} # copy IAM before each cut-over
STATE_DIR=${STATE_DIR:-/etc/honkio-s3/migration}
NGINX_SITE=${NGINX_SITE:-/etc/nginx/sites-available/honkio-s3-router.conf}
NGINX_BUCKETS=${NGINX_BUCKETS:-/etc/nginx/honkio-s3-router/buckets.conf}
LOG=${LOG:-/var/log/honkio-s3-cutover.log}
MC=${MC:-mc}
STATE=${STATE_DIR}/buckets # lines: "<bucket> <minio-frozen/honkio-s3/honkio-s3-frozen>"
TARGETS=${STATE_DIR}/targets # files: <bucket> = id of the Honkio S3 -> MinIO replication target
HERE=$(cd "$(dirname "$(readlink -f "$0")")" && pwd)

log() { echo "$(date -Is) $*" | tee -a "$LOG" >&2; }
die() { log "ERROR: $*"; exit 1; }
```

```

need_root() { [ "$(id -u)" = 0 ] || die "run as root"; }
lock() { exec 9>/run/honkio-s3-cutover.lock; flock -w 900 9 || die "another honkio-s3-cutover is running"; }
valid_bucket() { [[ "$1" =~ ^[a-z0-9][a-z0-9.-]{1,61}[a-z0-9]$ ]] || die "invalid bucket name: $1"; }

owner_of() { [ -f "$STATE" ] && awk -v b="$1" '$1==b {print $2}' "$STATE" || true; }
moved_buckets() { [ -f "$STATE" ] && awk '{print $1}' "$STATE" || true; }
set_owner() { # <bucket> <owner/minio>
    install -d -m 0755 "$STATE_DIR"
    local tmp; tmp=$(mktemp "$STATE_DIR/.buckets.XXXXXX")
    { [ -f "$STATE" ] && awk -v b="$1" '$1!=b' "$STATE"; [ "$2" = minio ] || echo "$1 $2"; } | sort >
"$tmp"
    chmod 0644 "$tmp"; mv "$tmp" "$STATE"
}

# Writes the nginx bucket routes from the state file and reloads nginx; the
# previous file is restored if nginx rejects the new one.
render_nginx() {
    install -d -m 0755 "$(dirname "$NGINX_BUCKETS")"
    local tmp b o; tmp=$(mktemp)
    {
        echo "# Generated by honkio-s3-cutover from $STATE. Do not edit."
        if [ -f "$STATE" ]; then while read -r b o; do
            local up=minio_s3_local freeze=""
            case "$o" in
                honkio-s3) up=honkio_s3_local ;;
                honkio-s3-frozen) up=honkio_s3_local; freeze=1 ;;
                minio-frozen) freeze=1 ;;
                *) continue ;;
            esac
            for loc in "= /$b" "^~ /$b/"; do
                echo "location $loc {"
                [ -n "$freeze" ] && echo '    if ($request_method !~ ^(GET|HEAD|OPTIONS)$) { return 503; }'

                echo "    proxy_pass http://$up;"
                echo "}"
            done
            done < "$STATE"; fi
        } > "$tmp"
        local backup=""
        [ -f "$NGINX_BUCKETS" ] && { backup=$(mktemp); cp "$NGINX_BUCKETS" "$backup"; }
        install -m 0644 "$tmp" "$NGINX_BUCKETS"; rm -f "$tmp"
        if ! nginx -t -q 2>/dev/null; then
            if [ -n "$backup" ]; then cp "$backup" "$NGINX_BUCKETS"; else : > "$NGINX_BUCKETS"; fi
            rm -f "$backup"
            die "nginx rejected the bucket routes; previous routes restored"
        fi
        rm -f "$backup"
        systemctl reload nginx
    }

router_active() { firewall-cmd --zone="$FIREWALL_ZONE" --query-forward-port="port=$PUBLIC_PORT:proto=tcp:toport=$ROUTER_PORT" >/dev/null 2>&1; }

router_ok() {
    [ "$(curl -s -m 5 -o /dev/null -w '%{http_code}' -H "Host: $HOST_HEADER" "http://127.0.0.1:$ROUTER_PORT/minio/health/live")" = 200 ]
}

server_of() { # <bucket>: which server answers for it through the router
    { curl -s -m 5 -o /dev/null -D - -H "Host: $HOST_HEADER" "http://127.0.0.1:$ROUTER_PORT/$1/" || true; }
} | tr -d '\r' | sed -n 's/^[Ss]erver: //p'
}

# Reads KEY=VALUE files (systemd EnvironmentFile format) without sourcing them.
env_value() { # <file> <key>
    python3 - "$1" "$2" <<'PY'
import sys
path, key = sys.argv[1], sys.argv[2]
for line in open(path, encoding="utf-8"):

```

```

    line = line.strip()
    if line.startswith(key + "="):
        v = line[len(key) + 1:]
        if len(v) >= 2 and v[0] == v[-1] and v[0] in "\"'":
            v = v[1:-1]
        print(v, end="")
        break
PY
}

# ----- console API

CTMP=""
console_login() {
    CTMP=$(mktemp -d); chmod 700 "$CTMP"
    python3 - "$HONKIO_S3_ENV" > "$CTMP/login.json" <<'PY'
import json, sys
env = {}
for line in open(sys.argv[1], encoding="utf-8"):
    if "=" in line and not line.lstrip().startswith("#"):
        k, v = line.strip().split("=", 1)
        env[k] = v[1:-1] if len(v) >= 2 and v[0] == v[-1] and v[0] in "\"'\" else v
print(json.dumps({"accessKey": env["MINIO_ROOT_USER"], "secretKey": env["MINIO_ROOT_PASSWORD"]}))
PY
    local code
    code=$(curl -s -o /dev/null -w '%{http_code}' -c "$CTMP/jar" -H 'x-honkio-s3-console: 1' -H 'content-type: application/json' \
        --data @"$CTMP/login.json" "$CONSOLE_URL/api/login")
    rm -f "$CTMP/login.json"
    [ "$code" = 200 ] || die "Honkio S3 console login failed ($code)"
}
console_logout() { [ -n "$CTMP" ] && { curl -s -o /dev/null -b "$CTMP/jar" -X POST -H 'x-honkio-s3-console: 1' "$CONSOLE_URL/api/logout" || true; rm -rf "$CTMP"; CTMP=""; }; }
capi() { # <method> <path> [json-file]
    local args=(-s -b "$CTMP/jar" -X "$1" -H 'x-honkio-s3-console: 1' -H 'content-type: application/json')
    [ $# -ge 3 ] && args+=(--data @"$3")
    curl "${args[@]}" "$CONSOLE_URL/api$2"
}

add_reverse_target() { # <bucket>: Honkio S3 replicates the bucket back to MinIO
    local b=$1
    python3 - "$MINIO_ENV" "$MINIO_URL" "$b" > "$CTMP/target.json" <<'PY'
import json, sys
env = {}
for line in open(sys.argv[1], encoding="utf-8"):
    if "=" in line and not line.lstrip().startswith("#"):
        k, v = line.strip().split("=", 1)
        env[k] = v[1:-1] if len(v) >= 2 and v[0] == v[-1] and v[0] in "\"'\" else v
print(json.dumps({"endpoint": sys.argv[2], "bucket": sys.argv[3], "accessKey": env["MINIO_ROOT_USER"],
    "secretKey": env["MINIO_ROOT_PASSWORD"], "replicateDeletes": True, "enabled": True,
    "createBucket": False}))
PY
    local out id
    out=$(capi POST "/buckets/$b/replication" "$CTMP/target.json"); rm -f "$CTMP/target.json"
    id=$(printf '%s' "$out" | python3 -c 'import json,sys; print(json.load(sys.stdin).get("id",""))'
2>/dev/null || true)
    [ -n "$id" ] || die "could not add the MinIO replication target for $b: $(printf '%s' "$out" | head -c
300)"
    install -d -m 0700 "$TARGETS"; echo "$id" > "$TARGETS/$b"
    log "$b: replication Honkio S3 -> MinIO added (target $id), waiting for the first resync"
    local i
    for i in $(seq 1 360); do
        out=$(capi GET "/buckets/$b/replication")
        if printf '%s' "$out" | python3 -c '
import json, sys
t = [x for x in json.load(sys.stdin)["targets"] if x["id"] == sys.argv[1]][0]["status"]
sys.exit(0 if t.get("lastFullSyncAt") else 1)' "$id" 2>/dev/null; then
            log "$b: reverse replication in sync"
            return 0
        fi
    fi
}

```

```

        sleep 5
    done
    die "$b: reverse replication did not finish its first resync in 30 min (check the console)"
}

remove_reverse_target() { # <bucket>
    local b=$1 id
    [ -f "$TARGETS/$b" ] || return 0
    id=$(cat "$TARGETS/$b")
    capi DELETE "/buckets/$b/replication/$id" >/dev/null || true
    rm -f "$TARGETS/$b"
    log "$b: replication Honkio S3 -> MinIO removed"
}

# ----- commands

copy_bucket_settings() { # <from-alias> <to-alias> <bucket>
    local s=$1 d=$2 b=$3 tmp
    tmp=$(mktemp -d)
    $MC anonymous get-json "$s/$b" > "$tmp/anon.json" 2>/dev/null || true
    if grep -q '"Effect"' "$tmp/anon.json"; then $MC anonymous set-json "$tmp/anon.json" "$d/$b"
>/dev/null; else $MC anonymous set private "$d/$b" >/dev/null 2>&1 || true; fi
    local q; q=$( $MC quota info --json "$s/$b" 2>/dev/null | sed -n 's/.*"quota":\([0-9]*\)*/\1/p' )
    if [ "$q:-0" -gt 0 ]; then $MC quota set "$d/$b" --size "$q" >/dev/null; fi
    if $MC ilm rule ls "$s/$b" >/dev/null 2>&1; then $MC ilm rule export "$s/$b" > "$tmp/ilm.json" && $MC
ilm rule import "$d/$b" < "$tmp/ilm.json" >/dev/null; fi
    rm -rf "$tmp"
}

verify() { "$HERE/verify-migration.sh" "$@"; }

cmd_iam_sync() {
    need_root
    local tmp user out
    tmp=$(mktemp -d); chmod 700 "$tmp"
    ( cd "$tmp" && $MC admin cluster iam export "$SRC_ALIAS" >/dev/null && unzip -q ./*.zip ) || { rm -rf
"$tmp"; die "IAM export from MinIO failed"; }
    chown -R "$HONKIO_S3_USER" "$tmp"
    user=$(env_value "$HONKIO_S3_ENV" MINIO_ROOT_USER)
    out=$(sudo -u "$HONKIO_S3_USER" env MINIO_ROOT_USER="$user" /usr/local/bin/honkio-s3 admin --data
"$HONKIO_S3_DATA" import-minio-iam "$tmp/iam-assets" 2>&1) || { rm -rf "$tmp"; die "IAM import failed:
$out"; }
    rm -rf "$tmp"
    log "iam-sync: $out"
}

cmd_cutover() {
    need_root; lock
    [ $# -gt 0 ] || die "usage: cutover <bucket>..."
    router_active || die "the router is off: run 'honkio-s3-cutover router-on' first"
    router_ok || die "the router does not answer on port $ROUTER_PORT"
    if [ "$CUTOVER_IAM_SYNC" = yes ]; then cmd_iam_sync; fi
    local b
    for b in "$@"; do
        valid_bucket "$b"
        [ -z "$(owner_of "$b")" ] || die "$b is already moved ($(owner_of "$b"))"
        $MC ls "$SRC_ALIAS/$b" >/dev/null 2>&1 || die "$b does not exist in MinIO"
        log "$b: cut-over starts (writes paused)"
        set_owner "$b" minio-frozen; render_nginx
        sleep 5 # Let running uploads finish
        systemctl restart "$SYNC_UNIT" # the sync now skips $b
        $MC mb --ignore-existing "$DST_ALIAS/$b" >/dev/null
        $MC mirror --overwrite --remove --preserve "$SRC_ALIAS/$b" "$DST_ALIAS/$b" >/dev/null
2>"$STATE_DIR/.mirror-$b.err" || true
        copy_bucket_settings "$SRC_ALIAS" "$DST_ALIAS" "$b"
        if ! verify "$SRC_ALIAS" "$DST_ALIAS" "$b" >"$STATE_DIR/.verify-$b.txt" 2>&1; then
            set_owner "$b" minio; render_nginx; systemctl restart "$SYNC_UNIT"
            die "$b: copies differ after the final mirror ($STATE_DIR/.verify-$b.txt); the bucket stays on
MinIO"
        fi
    done
}

```

```

rm -f "$STATE_DIR/.mirror-$b.err" "$STATE_DIR/.verify-$b.txt"
if [ "$REVERSE_REPLICATION" = yes ]; then
    console_login
    if ! ( add_reverse_target "$b" ); then
        remove_reverse_target "$b"; console_logout
        set_owner "$b" minio; render_nginx; systemctl restart "$SYNC_UNIT"
        die "$b: reverse replication failed; the bucket stays on MinIO"
    fi
    console_logout
fi
set_owner "$b" honkio-s3; render_nginx
# nginx reload is asynchronous: old workers answer briefly with the old route
local srv i; for i in 1 2 3 4 5 6 7 8 9 10; do srv=$(server_of "$b"); [ "$srv" = Honkio-S3 ] &&
break; sleep 1; done
[ "$srv" = Honkio-S3 ] || log "WARNING: $b answers as '$srv' through the router, expected Honkio-
S3"
log "$b: now served by Honkio S3 (writes open)"
done
}

cmd_rollback() {
    need_root; lock
    [ $# -gt 0 ] || die "usage: rollback <bucket>..."
    local b o
    for b in "$@"; do
        valid_bucket "$b"
        o=$(owner_of "$b")
        case "$o" in honkio-s3|honkio-s3-frozen|minio-frozen) ;; *) die "$b is not moved" ;; esac
        log "$b: rollback starts (writes paused)"
        if [ "$o" != minio-frozen ]; then
            set_owner "$b" honkio-s3-frozen; render_nginx
            sleep 5
            # Honkio S3 has been the source of truth: bring MinIO up to date.
            $MC mirror --overwrite --remove --preserve "$DST_ALIAS/$b" "$SRC_ALIAS/$b" >/dev/null 2>&1 ||
true
            copy_bucket_settings "$DST_ALIAS" "$SRC_ALIAS" "$b"
            verify "$DST_ALIAS" "$SRC_ALIAS" "$b" >/dev/null 2>&1 || die "$b: MinIO copy differs from
Honkio S3; bucket left on Honkio S3 with writes paused. Fix and re-run rollback."
            if [ -f "$TARGETS/$b" ]; then console_login; remove_reverse_target "$b"; console_logout; fi
        fi
        set_owner "$b" minio; render_nginx
        systemctl restart "$SYNC_UNIT" # the sync covers $b again
        log "$b: back on MinIO (writes open)"
    done
}

cmd_router_on() {
    need_root; lock
    [ -e "/etc/nginx/sites-enabled/$(basename "$NGINX_SITE")" ] || die "router not installed: run 'honkio-
s3-cutover install'"
    render_nginx
    router_ok || die "the router does not answer on port $ROUTER_PORT"
    if ! router_active; then
        firewall-cmd -q --zone="$FIREWALL_ZONE" --add-forward-
port="port=$PUBLIC_PORT:proto=tcp:toport=$ROUTER_PORT"
        firewall-cmd -q --permanent --zone="$FIREWALL_ZONE" --add-forward-
port="port=$PUBLIC_PORT:proto=tcp:toport=$ROUTER_PORT"
    fi
    log "router on: external traffic to port $PUBLIC_PORT goes through nginx :$ROUTER_PORT"
}

cmd_router_off() {
    need_root; lock
    if [ -n "$(moved_buckets)" ] && [ "${1:-}" != --force ]; then
        die "buckets are on Honkio S3 ($(moved_buckets | tr '\n' ' ')): roll them back first, or --force
(they would be served from MinIO's copy)"
    fi
    firewall-cmd -q --zone="$FIREWALL_ZONE" --remove-forward-
port="port=$PUBLIC_PORT:proto=tcp:toport=$ROUTER_PORT" || true
    firewall-cmd -q --permanent --zone="$FIREWALL_ZONE" --remove-forward-

```

```

port="port=$PUBLIC_PORT:proto=tcp:toport=$ROUTER_PORT" || true
log "router off: external traffic goes straight to MinIO"
}

cmd_status() {
    local b o
    echo "router:          $(router_active && echo ON || echo off) (forward $PUBLIC_PORT -> $ROUTER_PORT),
nginx $(router_ok && echo answering || echo NOT answering)"
    echo "sync:            $(systemctl is-active "$SYNC_UNIT" 2>/dev/null || true) ($SYNC_UNIT)"
    echo "iam-sync:         $(systemctl is-active honkio-s3-iam-sync.timer 2>/dev/null || true) (honkio-s3-
iam-sync.timer)"
    echo "moved buckets:"
    if [ -z "$(moved_buckets)" ]; then echo " (none)"; fi
    if [ -f "$STATE" ]; then while read -r b o; do
        printf ' %40s %-14s answers as %-8s reverse replication %s\n' "$b" "$o" "$(server_of "$b")" "$([
-f "$TARGETS/$b" ] && echo on || echo off)"
    done < "$STATE"; fi
    return 0
}

cmd_sync_run() {
    local args=() b
    for b in $(moved_buckets) $SYNC_EXCLUDE_BUCKETS; do args+=(--exclude-bucket "$b"); done
    if [ "$SYNC_WATCH" = yes ]; then args+=(--watch); fi
    exec $MC mirror --overwrite --remove --preserve --json --limit-upload "$SYNC_LIMIT" "${args[@]}"
"$SRC_ALIAS" "$DST_ALIAS"
}

cmd_install() {
    need_root
    install -m 0755 "$HERE/honkio-s3-cutover" /usr/local/sbin/honkio-s3-cutover
    install -m 0755 "$HERE/verify-migration.sh" /usr/local/sbin/verify-migration.sh
    install -d -m 0755 "$STATE_DIR" "$(dirname "$NGINX_BUCKETS")"
    [ -f "$NGINX_BUCKETS" ] || echo "# Generated by honkio-s3-cutover. Do not edit." > "$NGINX_BUCKETS"
    install -m 0644 "$HERE/nginx-honkio-s3-router.conf" "$NGINX_SITE"
    ln -sf "$NGINX_SITE" "/etc/nginx/sites-enabled/$(basename "$NGINX_SITE")"
    nginx -t -q && systemctl reload nginx
    install -m 0644 "$HERE/honkio-s3-sync.service" /etc/systemd/system/honkio-s3-sync.service
    install -m 0644 "$HERE/honkio-s3-iam-sync.service" "$HERE/honkio-s3-iam-sync.timer"
/etc/systemd/system/
systemctl daemon-reload
systemctl enable --now honkio-s3-iam-sync.timer >/dev/null
log "installed: router (nginx :$ROUTER_PORT, traffic not redirected yet), units,
/usr/local/sbin/honkio-s3-cutover"
echo "Next: 'systemctl restart $SYNC_UNIT' (new unit), then 'honkio-s3-cutover router-on'."
}

case "${1:-}" in
install) cmd_install ;;
status) cmd_status ;;
router-on) cmd_router_on ;;
router-off) shift; cmd_router_off "$@" ;;
cutover) shift; cmd_cutover "$@" ;;
rollback) shift; cmd_rollback "$@" ;;
iam-sync) need_root; lock; cmd_iam_sync ;;
verify) shift; if [ $# -gt 0 ]; then verify "$SRC_ALIAS" "$DST_ALIAS" "$@"; else verify "$SRC_ALIAS"
"$DST_ALIAS"; fi ;;
sync-run) cmd_sync_run ;;
*) sed -n '2,32p' "$0" | sed 's/^# \{0,1\}\/'; exit 2 ;;
esac

```

Appendix K: Migration router (nginx)

File in the repository: `deploy/migration/nginx-honkio-s3-router.conf`

```
# MinIO -> Honkio S3 migration router (docs/MIGRAATIO.md, "Siirto bucket kerrallaan").
# /etc/nginx/sites-available/honkio-s3-router.conf, enabled with a symlink in sites-enabled.
#
# Clients keep using the MinIO port: `honkio-s3-cutover router-on` redirects
# external traffic to port 9000 here (firewalld forward-port; local traffic to
# 127.0.0.1:9000 is not affected). Each bucket goes to MinIO unless
# /etc/nginx/honkio-s3-router/buckets.conf (written by honkio-s3-cutover) routes it
# to Honkio S3. Path-style addressing: /<bucket> and /<bucket>/<key>.

upstream minio_s3_local {
    server 127.0.0.1:9000;
    keepalive 32;
}

upstream honkio_s3_local {
    server 127.0.0.1:9300;
    keepalive 32;
}

server {
    listen 9080;
    server_name _;
    server_tokens off;
    access_log /var/log/nginx/honkio-s3-router.access.log;
    error_log /var/log/nginx/honkio-s3-router.error.log;

    # Same behaviour as a direct MinIO connection: no size limit, streaming
    # in both directions, special characters in headers allowed.
    ignore_invalid_headers off;
    client_max_body_size 0;
    proxy_buffering off;
    proxy_request_buffering off;
    proxy_max_temp_file_size 0;
    proxy_http_version 1.1;
    proxy_set_header Connection "";
    # SigV4 signs the Host header: pass it unchanged. X-Forwarded-* and
    # X-Real-IP from the edge proxy pass through as they are.
    proxy_set_header Host $http_host;
    # Keep the backend's own Server header (MinIO / Honkio-S3), not nginx's.
    proxy_pass_header Server;
    proxy_connect_timeout 300s;
    proxy_read_timeout 3600s;
    proxy_send_timeout 3600s;

    # Buckets moved to Honkio S3 (generated; may be empty).
    include /etc/nginx/honkio-s3-router/buckets.conf;

    location / {
        proxy_pass http://minio_s3_local;
    }
}
```

Appendix L: Migration sync service

File in the repository: `deploy/migration/honkio-s3-sync.service`

```
# Continuous MinIO -> Honkio S3 copy until the cut-over (docs/MIGRAATIO.md, step 5).
# /etc/systemd/system/honkio-s3-sync.service
#
# Needs root's mc aliases: "myminio" (MinIO, root credentials) and "honkio-s3"
# (Honkio S3, root credentials). Adjust --limit-upload to what the disks tolerate.
[Unit]
Description=Continuous MinIO -> Honkio S3 mirror (migration)
After=network-online.target minio.service honkio-s3.service

[Service]
Type=simple
User=root
Environment=HOME=/root
# --overwrite: changed objects are copied again; --remove: deletes follow;
# --preserve: keep content type and metadata; --json: machine-readable log.
# honkio-s3-cutover sync-run = mc mirror --watch --overwrite --remove --preserve --json
# --limit-upload $SYNC_LIMIT --exclude-bucket <each bucket already moved to Honkio S3> myminio honkio-s3
# (settings in /etc/honkio-s3/migration.conf; without the tool, run that mc command directly)
ExecStart=/usr/local/sbin/honkio-s3-cutover sync-run
Restart=always
RestartSec=30
# The copy must not slow down the production MinIO.
Nice=19
IOSchedulingClass=idle
MemoryMax=512M

[Install]
WantedBy=multi-user.target
```

Appendix M: IAM sync service

File in the repository: `deploy/migration/honkio-s3-iam-sync.service`

```
# Copies MinIO users and policies to Honkio S3 during the migration
# (honkio-s3-cutover iam-sync). /etc/systemd/system/honkio-s3-iam-sync.service
[Unit]
Description=Copy MinIO IAM (users, policies) to Honkio S3
After=minio.service honkio-s3.service

[Service]
Type=oneshot
Environment=HOME=/root
ExecStart=/usr/local/sbin/honkio-s3-cutover iam-sync
Nice=10
```

Appendix N: IAM sync timer

File in the repository: `deploy/migration/honkio-s3-iam-sync.timer`

```
# /etc/systemd/system/honkio-s3-iam-sync.timer
[Unit]
Description=Copy MinIO IAM to Honkio S3 every 10 minutes (migration)

[Timer]
OnBootSec=5min
OnUnitActiveSec=10min
RandomizedDelaySec=30

[Install]
WantedBy=timers.target
```

Appendix O: Migration check script

File in the repository: `deploy/migration/verify-migration.sh`

```
#!/bin/sh
# Compares every bucket of a MinIO alias with a Honkio S3 alias: object count,
# total size and `mc diff` (missing / extra / different objects).
# Read-only: Lists objects on both sides, changes nothing.
#
# Usage: verify-migration.sh <source-alias> <target-alias> [bucket ...]
#   verify-migration.sh myminio honkio-s3           # all buckets of the source
#   verify-migration.sh myminio honkio-s3 photos     # only these buckets
# Exit status 0 = all buckets identical, 1 = differences found.
set -u
SRC=${1:?source alias}
DST=${2:?target alias}
shift 2
MC=${MC:-mc}

if [ $# -gt 0 ]; then
    BUCKETS="$*"
else
    BUCKETS=$(MC ls --json "$SRC" | sed -n 's/.*"key": "\([^"]*\)\/".*\/1/p')
fi

du_field() { # <alias/bucket> <field>
    MC du --json "$1" 2>/dev/null | sed -n "s/.*\"$2\": \"([0-9]*).\".*\/1/p" | head -1
}

bad=0
printf '%-40s %12s %12s %16s %16s %8s\n' BUCKET SRC_OBJECTS DST_OBJECTS SRC_BYTES DST_BYTES DIFFS
for b in $BUCKETS; do
    so=$(du_field "$SRC/$b" objects); do_=$(du_field "$DST/$b" objects)
    sb=$(du_field "$SRC/$b" size); db=$(du_field "$DST/$b" size)
    diffs=$(MC diff "$SRC/$b" "$DST/$b" 2>&1 | wc -l)
    mark=""
    if [ "${so:-x}" != "${do_:-y}" ] || [ "${sb:-x}" != "${db:-y}" ] || [ "$diffs" -ne 0 ]; then
        mark=" <-- DIFFERS"
        bad=1
    fi
    printf '%-40s %12s %12s %16s %16s %8s\n' "$b" "${so:--}" "${do_:--}" "${sb:--}" "${db:--}" "$diffs"
    "$mark"
done

if [ $bad -eq 0 ]; then
    echo "OK: all buckets identical"
else
    echo "DIFFERENCES: run 'mc diff $SRC/<bucket> $DST/<bucket>' for details"
    echo "(< only in $SRC, > only in $DST, ! size differs)"
fi
exit $bad
```

Appendix P: Migration tool self-test

File in the repository: `deploy/migration/test-honkio-s3-cutover.sh`

```
#!/bin/bash
# Self-test of honkio-s3-cutover on a scratch setup. Run as root on the migration
# host BEFORE using the tool for real (needs nginx, firewallld, mc, honkio-s3).
#
# Everything is temporary and separate from production: two Honkio S3 servers
# play "MinIO" (127.0.0.1:19001) and "Honkio S3" (19002, console 19003), the
# router listens on 19080 and the test forward-port is 19000 -> 19080.
# The test nginx site, sync unit and firewall rule are removed at the end.
#
# Usage: deploy/migration/test-honkio-s3-cutover.sh
set -euo pipefail
HERE=$(cd "$(dirname "$(readlink -f "$0")")" && pwd)
TOOL="$HERE/honkio-s3-cutover"
[ "${id -u}" = 0 ] || { echo "run as root"; exit 1; }
if [ -e /etc/nginx/sites-enabled/honkio-s3-router.conf ]; then
    echo "the production router is installed; its upstream names would clash with the test site"; exit 1
fi

T=$(mktemp -d /tmp/mxmig.XXXXXX); chmod 700 "$T"
PW=$(head -c 18 /dev/urandom | od -An -tx1 | tr -d ' \n')
SITE=/etc/nginx/sites-available/honkio-s3-router-test.conf
UNIT=/etc/systemd/system/honkio-s3-sync-test.service
PIDS=()
fails=0
check() { if eval "$2"; then echo "PASS $1"; else echo "FAIL $1"; fails=$((fails + 1)); fi; }
wait_for() { local i; for i in $(seq 1 "$2"); do if eval "$1"; then return 0; fi; sleep 1; done; return 1; }

cleanup() {
    set +e
    HONKIO_S3_CUTOVER_CONF=$T/migration.conf "$TOOL" router-off --force >/dev/null 2>&1
    systemctl stop honkio-s3-sync-test.service >/dev/null 2>&1; rm -f "$UNIT"; systemctl daemon-reload;
    systemctl reset-failed honkio-s3-sync-test.service >/dev/null 2>&1
    rm -f /etc/nginx/sites-enabled/honkio-s3-router-test.conf "$SITE"; nginx -t -q 2>/dev/null &&
    systemctl reload nginx
    for p in "${PIDS[@]"; do kill "$p" 2>/dev/null; done
    sleep 1; rm -rf "$T"
}
trap cleanup EXIT

# --- scratch servers -----
mkdir -p "$T/src" "$T/dst" "$T/state" "$T/nginx" "$T/mc"
MINIO_ROOT_USER=tstroot MINIO_ROOT_PASSWORD=$PW HONKIO_S3_SCRUB_INTERVAL_DAYS=0 \
    /usr/local/bin/honkio-s3 server "$T/src" --address 127.0.0.1:19001 >"$T/src.log" 2>&1 & PIDS+=(!)
MINIO_ROOT_USER=tstroot MINIO_ROOT_PASSWORD=$PW HONKIO_S3_SCRUB_INTERVAL_DAYS=0
HONKIO_S3_REPLICATION_RESYNC_RATE_MB=0 \
    HONKIO_S3_CONSOLE_ADDRESS=127.0.0.1:19003 /usr/local/bin/honkio-s3 server "$T/dst" --address
127.0.0.1:19002 >"$T/dst.log" 2>&1 & PIDS+=(!)
printf 'MINIO_ROOT_USER=tstroot\nMINIO_ROOT_PASSWORD=%s\n' "$PW" > "$T/root.env"
printf
'MC_CONFIG_DIR=%s\nMC_HOST_tsrc=http://tstroot:%s@127.0.0.1:19001\nMC_HOST_tdst=http://tstroot:%s@127.0.0.
1:19002\nMC_HOST_trouter=http://tstroot:%s@127.0.0.1:19080\n' \
    "$T/mc" "$PW" "$PW" "$PW" > "$T/mc.env"
chmod 600 "$T"/*.env
set -a; . "$T/mc.env"; set +a
cat > "$T/migration.conf" <<EOF
SRC_ALIAS=tsrc
DST_ALIAS=tdst
MINIO_URL=http://127.0.0.1:19001
HONKIO_S3_URL=http://127.0.0.1:19002
CONSOLE_URL=http://127.0.0.1:19003
HONKIO_S3_ENV=$T/root.env
MINIO_ENV=$T/root.env
```

```

PUBLIC_PORT=19000
ROUTER_PORT=19080
HOST_HEADER=migration-test.invalid
SYNC_UNIT=honkio-s3-sync-test.service
SYNC_LIMIT=50MiB
SYNC_WATCH=no
STATE_DIR=$T/state
NGINX_SITE=$SITE
NGINX_BUCKETS=$T/nginx/buckets.conf
LOG=$T/migrate.log
CUTOVER_IAM_SYNC=no
EOF
export HONKIO_S3_CUTOVER_CONF=$T/migration.conf
sleep 2

# --- test data and the initial copy -----
mc mb -q tsrc/alpha >/dev/null; mc mb -q tsrc/beta >/dev/null
for i in 1 2 3; do echo "alpha $i" | mc pipe -q "tsrc/alpha/public/a$i.txt" >/dev/null; done
echo "beta" | mc pipe -q tsrc/beta/b.txt >/dev/null
echo '{"Version":"2012-10-17","Statement":[{"Effect":"Allow","Principal":{"AWS":["*"]},"Action":["s3:GetObject"],"Resource":["arn:aws:s3:::alpha/public/*"]}]}' > "$T/anon.json"
mc anonymous set-json "$T/anon.json" tsrc/alpha >/dev/null
mc mirror -q --overwrite --preserve tsrc tdst >/dev/null

# --- router (test copy of the production site, other ports) -----
sed -e 's/127.0.0.1:9000/127.0.0.1:19001/; s/127.0.0.1:9300/127.0.0.1:19002/; s/listen 9080;/listen 19080;/\ \
-e "s#/etc/nginx/honkio-s3-router/buckets.conf#$T/nginx/buckets.conf#" \
-e "s#/var/log/nginx/honkio-s3-router#$T/router#g" "$HERE/nginx-honkio-s3-router.conf" > "$SITE"
: > "$T/nginx/buckets.conf"
ln -sf "$SITE" /etc/nginx/sites-enabled/honkio-s3-router-test.conf
nginx -t -q && systemctl reload nginx

cat > "$UNIT" <<EOF
[Service]
EnvironmentFile=$T/mc.env
Environment=HONKIO_S3_CUTOVER_CONF=$T/migration.conf
ExecStart=$TOOL sync-run
# The scratch "MinIO" is a Honkio S3, which has no bucket change notifications for
# `mc mirror --watch`: run full passes back to back instead.
Restart=always
RestartSec=2
EOF
systemctl daemon-reload
systemctl start honkio-s3-sync-test.service

"$TOOL" router-on
check "router forwards the public test port" "firewall-cmd --query-forward-port=port=19000:proto=tcp:toport=19080 >/dev/null"
check "router answers" "[ \"\$(curl -s -o /dev/null -w '%{http_code}' http://127.0.0.1:19080/minio/health/live)\"] = 200 ]"
check "anonymous read through the router (MinIO side)" "[ \"\$(curl -s http://127.0.0.1:19080/alpha/public/a1.txt)\"] = 'alpha 1' ]"

# --- cut-over -----
"$TOOL" cutover alpha
check "state: alpha on Honkio S3" "grep -q '^alpha honkio-s3$' $T/state/buckets"
check "nginx routes alpha to Honkio S3" "grep -q 'proxy_pass http://honkio_s3_local' $T/nginx/buckets.conf"
echo "new" | mc pipe -q trouter/alpha/public/new.txt >/dev/null
check "write through the router lands in Honkio S3" "mc stat tdst/alpha/public/new.txt >/dev/null 2>&1"
check "Honkio S3 replicates it back to MinIO" "wait_for 'mc stat tsrc/alpha/public/new.txt >/dev/null 2>&1' 20"
check "anonymous read of a moved bucket" "[ \"\$(curl -s http://127.0.0.1:19080/alpha/public/new.txt)\"] = new ]"
echo "beta2" | mc pipe -q trouter/beta/b2.txt >/dev/null
check "other buckets still go to MinIO" "mc stat tsrc/beta/b2.txt >/dev/null 2>&1"
check "and the sync still copies them" "wait_for 'mc stat tdst/beta/b2.txt >/dev/null 2>&1' 30"
echo "direct" | mc pipe -q tsrc/alpha/direct.txt >/dev/null
sleep 8

```

```

check "the sync skips the moved bucket" "! mc stat tdst/alpha/direct.txt >/dev/null 2>&1"
mc rm -q tsrc/alpha/direct.txt >/dev/null
check "the moved bucket answers as Honkio S3" "\"$TOOL\" status | grep -q 'alpha .*honkio-s3 .*answers as Honkio-S3'"
check "status lists the bucket" "\"$TOOL\" status | grep -q 'alpha .*honkio-s3'"
check "router-off refuses while buckets are moved" "! \"$TOOL\" router-off >/dev/null 2>&1"

# --- rollback -----
echo "after" | mc pipe -q trouter/alpha/public/after.txt >/dev/null
"$TOOL" rollback alpha
check "state: alpha back on MinIO" "! grep -q '^alpha' $T/state/buckets"
check "objects written on Honkio S3 are in MinIO" "mc stat tsrc/alpha/public/after.txt >/dev/null 2>&1 && mc stat tsrc/alpha/public/new.txt >/dev/null 2>&1"
echo "back" | mc pipe -q trouter/alpha/back.txt >/dev/null
check "writes go to MinIO again" "mc stat tsrc/alpha/back.txt >/dev/null 2>&1"
check "and the sync copies the bucket again" "wait_for 'mc stat tdst/alpha/back.txt >/dev/null 2>&1' 30"

"$TOOL" router-off
check "router off" "! firewall-cmd --query-forward-port=port=19000:proto=tcp:toport=19080 >/dev/null"

echo
if [ $fails -eq 0 ]; then
    echo "ALL PASSED"
else
    echo "$fails FAILED"; cat "$T/migrate.log"
    echo "--- sync unit"; journalctl -u honkio-s3-sync-test.service -n 15 --no-pager -o cat | cut -c1-300
fi
exit $fails

```