Disaster Recovery Runbook

Disaster Recovery Runbook

Status: LIVE — restore commands below verified against the config/scripts actually running on 2026-07-28. See ALAI Backup Strategy for the architecture these steps restore from.

IMPORTANT — read before following any restore procedure: `~/system/architecture/litestream-restore-runbook.md` (dated 2026-04-20) documents `litestream restore ... abs://alaibackups0ebb/...` commands. Those commands target the **Azure Blob replica path, which is no longer being written to** — as of the CEO cost decision on 2026-07-13 (MC #105462), all 64 litestream-managed databases replicate to **local file replicas** only (`~/backups/litestream/<db>/`), confirmed live in `~/system/config/litestream.yml` on 2026-07-28. Any data in `alaibackups0ebb` predates 2026-07-13 and will be stale. Use the file-replica commands in Layer 2 below, not the old `abs://` commands, unless the Azure Blob SQLite path has since been reactivated (check §6 of the Backup Strategy page first).

Layer 1 — Git (fastest, use first for any code/config loss)

Source of truth: GitHub/origin remote for each repo (`~/system`, `~/ALAI`, `~/.claude`, `~/projects/*/*`).

```
# Clone fresh
git clone <remote-url> <target-dir>

# Or, if the local repo exists but is behind/corrupted:
cd <repo>
git fetch origin
git reset --hard origin/<branch>    # DESTRUCTIVE — confirm no uncommitted work worth keeping first
```

Recovery point objective (RPO): up to 1 hour of uncommitted work (hourly-backup.sh cron interval, `0 * * * *`). Check `~/system/logs/hourly-backup-cron.log` for the last successful push timestamp before relying on this.

Layer 2 — SQLite databases

2a. Normal case — restore from local litestream replica (PRIMARY, live today)

```
# Example: mission-control.db
litestream restore \
  -o /tmp/mission-control-restored.db \
  ~/backups/litestream/mission-control

sqlite3 /tmp/mission-control-restored.db "PRAGMA integrity_check;"
sqlite3 /tmp/mission-control-restored.db "SELECT COUNT(*) FROM tasks;"

# If good, replace the live DB (stop anything writing to it first):
cp /tmp/mission-control-restored.db ~/system/databases/mission-control.db
```

Repeat per-database using the replica path from `~/system/config/litestream.yml` (`replicas[].path`, all currently under `~/backups/litestream/<db>/`). Lag from primary under normal operation: sub-minute (1s–300s depending on DB tier — P0-critical/financial sync fastest).

2b. ANVIL local disk lost entirely — restore from Backblaze B2 offsite

```
# List what's there
rclone lsd b2-alai:alai-studio-backup/system-databases/

# Pull a specific DB snapshot down
rclone copy b2-alai:alai-studio-backup/system-databases/mission-control.db /tmp/restore/

# Or pull the full daily snapshot set
rclone copy b2-alai:alai-studio-backup/databases/ /tmp/restore-daily/
```

Two independent B2 jobs exist — `rclone-backup.sh` (daily 03:00, `system-databases/` + `claude-memory/`) and `offsite-backup.sh` (every 6h, `databases/` + `config/` + `rules/` + `specs/` + `tools/` + `.claude/*`). Check both prefixes; freshness may differ by up to 6h between them.

Caveat: B2 snapshots are point-in-time copies of the litestream local replica at sync time, not continuously replicated — worst-case data loss window is the sync interval of whichever job ran last (up to 6h), not the sub-minute litestream RPO.

2c. If the Azure Blob SQLite path has been reactivated since 2026-07-28

Only if `~/system/config/litestream.yml` shows `type: abs` replicas again (verify before using):

```
export AZURE_CLIENT_ID="1a0b3018-0c31-474b-918f-531b0a29a669"
export AZURE_CLIENT_SECRET="<retrieve from Bitwarden: alai-backup-writer>"
export AZURE_TENANT_ID="3454a03f-20b4-4bda-a116-2293c459aecd"

litestream restore \
  -o /tmp/mission-control-restored.db \
  abs://alaibackups0ebb/system-db-backups/litestream/mission-control
```

Full detail (promotion-to-write-primary on a replacement host, multi-scenario): `~/system/architecture/litestream-restore-runbook.md` — usable once the `abs://` path is confirmed live again; do not follow it blindly today.

Layer 3 — LightRAG / Neo4j Docker volumes

Source of truth (corrected 2026-08-08, MC #106946): this Mac (Makinja-sin-Mac-Studio), not the Azure VM — live since the 2026-08-03 migration (#106747). Backup script runs entirely local `docker run` / `docker compose`, no SSH involved.

Full procedure with all 3 restore scenarios (this machine, from Azure Blob offsite download, throwaway-volume verification) is maintained in LightRAG Backup runbook — do not duplicate here, it is live and current. Summary:

```
# 1. Pick snapshot (local safety net or download from Azure Blob first)
SNAPSHOT=~/system/backups/lightrag/<timestamp>
cd "$SNAPSHOT" && shasum -a 256 -c MANIFEST.sha256

# 2. Restore all 7 volumes (post-migration live names, lr106747-* prefix — MC #106946)
for vol in lr106747-data lr106747-kg lr106747-cache lr106747-neo4j-data lr106747-neo4j-import
```

```
lr106747-neo4j-logs lr106747-neo4j-plugins; do
  docker volume rm $vol || true
  docker volume create $vol
  docker run --rm -v $vol:/dst -v "$SNAPSHOT":/src alpine tar xzf /src/${vol}.tar.gz -C /dst
done

cd ~/system/lightrag-local && docker compose -f docker-compose.local.yml up -d
curl http://localhost:9621/health   # expect {"status":"healthy"}
```

To pull from the Azure offsite copy instead of the local safety net:

```
source ~/system/config/azure-lightrag-backup.env
az storage blob download-batch \
  --account-name $AZURE_STORAGE_ACCOUNT --account-key "$AZURE_STORAGE_KEY" \
  --source $AZURE_STORAGE_CONTAINER --destination ~/system/backups/lightrag/azure-restore-<TS>
\
  --pattern "<TS>/*"
```

Layer 4 — Full ANVIL loss (new Mac, everything gone)

No single automated bootstrap script covers this end-to-end today; `~/system/scripts/azure-blob-bootstrap.sh` and `~/system/scripts/migrate-lightrag-to-azure.sh` exist but were written against the April-plan Azure Blob SQLite path, which is not the live path (see Layer 2 note above) — treat them as reference, not a turnkey script, until re-validated.

Manual sequence:

1. **Provision** a new Mac (or VM), install prerequisites: `brew install git sqlite3 rclone litestream docker`.
2. **Layer 1 — Git:** clone `~/system`, `~/ALAI`, `~/.claude`, and every `~/projects/*/*` repo from their GitHub remotes.
3. **Layer 2 — SQLite:** restore each database from Backblaze B2 (§2b — this is the only surviving copy if ANVIL's local disk, including the litestream local replicas under `~/backups/litestream/`, is gone).
4. **Layer 3 — LightRAG/Neo4j:** the Azure VM (`vm-alai-lightrag`, 20.240.61.67) is a separate host from ANVIL — if only ANVIL is lost, this layer is untouched. If the Azure VM is also lost, restore from `plockfrontstaging`/`lightrag-backup` per §Layer 3 above onto a freshly provisioned VM.

5. **Reconfigure launchd:** re-load LaunchAgents for `com.alai.litestream`, `com.john.rclone-backup`, `com.john.offsite-backup`, `com.alai.lightrag-backup` (all found under `~/Library/LaunchAgents/` in the repo you just restored — none are currently `.disabled` except the Azure-Blob-SQLite ones, which stay disabled per the CEO cost decision unless explicitly reactivated).
6. **Verify:** `crontab -l` shows the 3 cron jobs (gotcha-health, db-backup, hourly-backup); `launchctl list | grep -E "litestream|backup"` shows all 4 active LaunchAgents loaded; `curl http://localhost:9621/health` (or the Azure VM equivalent) returns healthy.

This layer has **not been drill-tested end-to-end** — no MC evidence of a live full-ANVIL-loss rehearsal was found this session. Flagging per ZAKON PLAN: a Proveo/Angie Jones validation task simulating this (stop services, wipe a scratch volume, restore, verify) should exist before this runbook is trusted under real incident pressure.

Document Owner: Lexicon (Skillforge role) **Last Verified:** 2026-07-28 — cross-checked against `~/system/config/litestream.yml` (live, 64/64 `type: file`), live `litestream replicate` process, `crontab -l`, `launchctl list`, and `~/system/docs/runbooks/lightrag-backup.md`.

Revision #2
Created 2026-07-28 06:57:07 UTC by John
Updated 2026-08-08 13:37:35 UTC by John