Backup

Backup and restore runbooks for ALAI infrastructure.

- Azure DB Backup Extension (2026-04-20)

Azure DB Backup Extension (2026-04-20)

Azure DB Backup Extension — Postgres + Qdrant Targets (2026-04-20)

BookStack location: Infrastructure → Backup
Owner: Skillforge / FlowForge
Source of truth verified: `~/system/daemons/azure-db-backup.sh`, `~/projects/alai-system/config/launchagents/com.alai.azure-db-backup.plist`, archived runtime plist at `~/Library/LaunchAgents/_archive/2026-07/com.alai.azure-db-backup.plist.disabled`.
Last verified: 2026-07-28.

Current status

This page documents the Azure DB Backup extension that was added on 2026-04-20.

Current machine state is intentionally conservative:

- Script exists: `~/system/daemons/azure-db-backup.sh`.
- LaunchAgent spec exists in repo: `~/projects/alai-system/config/launchagents/com.alai.azure-db-backup.plist`.
- Runtime LaunchAgent is archived/disabled on this host: `~/Library/LaunchAgents/_archive/2026-07/com.alai.azure-db-backup.plist.disabled`.
- No active `launchctl list` entry for `com.alai.azure-db-backup` was present during verification.
- The target containers `drop-postgres-1`, `dropsrbija-postgres`, and `qdrant` were not running during verification; the script will warn-and-skip the Postgres targets, and Qdrant falls back only if it can inspect a `qdrant` container mount.

Azure destination

The script reads `~/system/config/azure-backup.env` and uploads to:

- Storage account: `alaibackups0ebb`
- Container: `system-db-backups`
- Auth mode in upload/download calls: `--auth-mode login` after isolated service-principal login

Do not expose or copy secrets from `azure-backup.env` into docs or tickets.

LaunchAgent

Label: `com.alai.azure-db-backup`

Program:

```
<array>
  <string>/bin/bash</string>
  <string>/Users/makinja/system/daemons/azure-db-backup.sh</string>
</array>
```

Schedule:

```
<key>StartInterval</key>
<integer>14400</integer>
<key>RunAtLoad</key>
<false/>
```

`14400` seconds = every 4 hours.

Logs:

- stdout: `/Users/makinja/system/logs/azure-db-backup.out`
- stderr: `/Users/makinja/system/logs/azure-db-backup.err`
- script log: `~/system/logs/azure-db-backup.log`

New backup targets added by the extension

Target	Source	Dump/snapshot method	Blob prefix

`postgres-drop`	Docker container `drop-postgres-1`, DB user `drop`	`docker exec drop-postgres-1 pg_dumpall -U drop	gzip`
`postgres-dropsrbija`	Docker container `dropsrbija-postgres`, DB user `dropsrbija`	`docker exec dropsrbija-postgres pg_dumpall -U dropsrbija	gzip`
`qdrant`	Qdrant API at `http://localhost:6333`	Collection snapshot API; fallback is paused-container volume/bind tar	`qdrant/`

Blob naming scheme

The script sets `DATE=$(date +%Y-%m-%d)` for each run.

Postgres blobs:

```
postgres-drop/$DATE/drop-postgres-1-$DATE.sql.gz
postgres-dropsrbija/$DATE/dropsrbija-postgres-$DATE.sql.gz
```

Qdrant API snapshot blobs:

```
qdrant/$DATE/$COLLECTION-$DATE.tar
```

Qdrant fallback volume blob:

```
qdrant/$DATE/qdrant-volume-$DATE.tar.gz
```

Run manifest:

```
MANIFEST-$DATE.txt
```

SHA-256 sidecar pattern

`upload_blob()` applies the same integrity pattern to every uploaded target:

1. Hash the local backup before upload:

   ```
   shasum -a 256 "$LOCAL_FILE"
   ```

2. Write a sidecar in `shasum -c` compatible format:

```
<sha256>  <filename>
```

3. Upload the backup blob.
4. Upload the sidecar as the same blob name with `.sha256` appended.
5. Download the uploaded blob back to `/tmp` and re-hash it.
6. Count the backup as successful only if pre-upload and post-download hashes match.

On mismatch the script logs `ERROR: SHA-256 MISMATCH`, increments the failure counter, and after `MAX_FAILURES=3` consecutive failed runs sends a Slack `#ops` alert and calls `alert-gate.js escalate azure-db-backup`.

Common restore preflight — download and verify first

Set the backup date and storage values before running target-specific restore commands:

```
DATE="2026-04-20"
ACCOUNT="alaibackups0ebb"
CONTAINER="system-db-backups"
```

Download and verify one blob:

```
BLOB="postgres-drop/$DATE/drop-postgres-1-$DATE.sql.gz"
FILE="/tmp/drop-postgres-1-$DATE.sql.gz"

az storage blob download --account-name "$ACCOUNT" --container-name "$CONTAINER" \
  --name "$BLOB" --file "$FILE" --auth-mode login
az storage blob download --account-name "$ACCOUNT" --container-name "$CONTAINER" \
  --name "$BLOB.sha256" --file "$FILE.sha256" --auth-mode login
(cd /tmp && shasum -a 256 -c "$(basename "$FILE").sha256")
```

Proceed only after `shasum -c` reports `OK`.

Restore: postgres-drop

The backup is `pg_dumpall` SQL compressed with gzip. Restore with `psql`, not `pg_restore`:

```
DATE="2026-04-20"
ACCOUNT="alaibackups0ebb"
CONTAINER="system-db-backups"
BLOB="postgres-drop/$DATE/drop-postgres-1-$DATE.sql.gz"
FILE="/tmp/drop-postgres-1-$DATE.sql.gz"

az storage blob download --account-name "$ACCOUNT" --container-name "$CONTAINER" \
  --name "$BLOB" --file "$FILE" --auth-mode login
az storage blob download --account-name "$ACCOUNT" --container-name "$CONTAINER" \
  --name "$BLOB.sha256" --file "$FILE.sha256" --auth-mode login
(cd /tmp && shasum -a 256 -c "$(basename "$FILE").sha256")

gunzip -c "$FILE" | docker exec -i drop-postgres-1 psql -U drop
```

Restore: postgres-dropsrbija

```
DATE="2026-04-20"
ACCOUNT="alaibackups0ebb"
CONTAINER="system-db-backups"
BLOB="postgres-dropsrbija/$DATE/dropsrbija-postgres-$DATE.sql.gz"
FILE="/tmp/dropsrbija-postgres-$DATE.sql.gz"

az storage blob download --account-name "$ACCOUNT" --container-name "$CONTAINER" \
  --name "$BLOB" --file "$FILE" --auth-mode login
az storage blob download --account-name "$ACCOUNT" --container-name "$CONTAINER" \
  --name "$BLOB.sha256" --file "$FILE.sha256" --auth-mode login
(cd /tmp && shasum -a 256 -c "$(basename "$FILE").sha256")

gunzip -c "$FILE" | docker exec -i dropsrbija-postgres psql -U dropsrbija
```

Restore: qdrant API snapshot

The API backup path creates a per-collection Qdrant snapshot and uploads it as:

```
qdrant/$DATE/$COLLECTION-$DATE.tar
```

Download and verify the snapshot first:

```
DATE="2026-04-20"
COLLECTION="collection-name"
ACCOUNT="alaibackups0ebb"
CONTAINER="system-db-backups"
BLOB="qdrant/$DATE/$COLLECTION-$DATE.tar"
FILE="/tmp/$COLLECTION-$DATE.tar"

az storage blob download --account-name "$ACCOUNT" --container-name "$CONTAINER" \
  --name "$BLOB" --file "$FILE" --auth-mode login
az storage blob download --account-name "$ACCOUNT" --container-name "$CONTAINER" \
  --name "$BLOB.sha256" --file "$FILE.sha256" --auth-mode login
(cd /tmp && shasum -a 256 -c "$(basename "$FILE").sha256")
```

Recovery must use the Qdrant snapshot-recovery API for the deployed Qdrant version. Verify the exact endpoint against that live Qdrant version before production restore; this session verified the backup script path, not a live Qdrant recovery endpoint.

Restore: qdrant fallback volume tar

If the backup was created by the script fallback path, it is stored as:

```
qdrant/$DATE/qdrant-volume-$DATE.tar.gz
```

Download, verify, then restore into the Qdrant data volume:

```
DATE="2026-04-20"
ACCOUNT="alaibackups0ebb"
CONTAINER="system-db-backups"
QDRANT_VOLUME="qdrant_storage"
BLOB="qdrant/$DATE/qdrant-volume-$DATE.tar.gz"
FILE="/tmp/qdrant-volume-$DATE.tar.gz"

az storage blob download --account-name "$ACCOUNT" --container-name "$CONTAINER" \
  --name "$BLOB" --file "$FILE" --auth-mode login
az storage blob download --account-name "$ACCOUNT" --container-name "$CONTAINER" \
  --name "$BLOB.sha256" --file "$FILE.sha256" --auth-mode login
(cd /tmp && shasum -a 256 -c "$(basename "$FILE").sha256")

docker pause qdrant
```

```
docker run --rm -v "$QDRANT_VOLUME:/dst" -v /tmp:/src alpine \
  sh -c "rm -rf /dst/* && tar xzf /src/qdrant-volume-$DATE.tar.gz -C /dst"
docker unpause qdrant
```

Confirm `QDRANT_VOLUME` from the live container before running the restore command:

```
docker inspect qdrant --format '{{range .Mounts}}{{if eq .Type
"volume"}}{{.Name}}{{end}}{{end}}'
```

Verification commands used for this page

```
bash -n ~/system/daemons/azure-db-backup.sh
plutil -lint ~/projects/alai-system/config/launchagents/com.alai.azure-db-backup.plist
plutil -lint ~/Library/LaunchAgents/_archive/2026-07/com.alai.azure-db-backup.plist.disabled
launchctl list | grep -F 'com.alai.azure-db-backup' || true
find ~/Library/LaunchAgents -path '*com.alai.azure-db-backup.plist*' -maxdepth 4 -print
docker ps --format '{{.Names}}' | egrep '^(drop-postgres-1|dropsrbija-postgres|qdrant)$' ||
true
```

Related local docs

- `~/system/docs/runbooks/azure-blob-offsite-backup-setup.md`
- `~/system/docs/runbooks/alai-backup-strategy.md`
- `~/system/docs/runbooks/lightrag-backup.md`