

Data

The dataset pin

🕒 3 min read · Updated 27 September 2026 · Verified against [aced1a0](#) on 2026-09-27

How this platform obtains the scripture database: `dataset.lock.json` names a sggs-data commit plus the object's sha256 and size; make dataset installs exactly that, verified; a new dataset is a bump.

<https://docs.gurbanisoul.com/data/dataset-pin/>

The database is never committed to this repository and never copied by hand. `dataset.lock.json` names the object; `scripts/data/fetch_dataset.py` is the only way it arrives — in CI, in the API image build and locally — streamed from sggs-data's Git LFS, verified against the lock, installed atomically. What production serves is exactly what sggs-data proved, byte for byte ([ADR-0008](#)).

The lock

`dataset.lock.json` · lines 1–10

```
{
  "repository": "Algorythmos-AI/sggs-data",
  "commit": "efbf74e4f018a3f95801af3f641feecd2042e348",
  "dataset_version": "1.3.0",
  "database": {
    "path": "db/sggs.sqlite",
    "sha256": "4e6777c54140d7c4ff76fd06b5453991daf4cabcb2c98479d216908d4c31455b",
    "size": 108625920
  }
}
```

Read from the source at build time — [view these lines on GitHub at c3d9360](#) —.

Source: **`dataset.lock.json`**² at the root of this repository.

`commit` is the sggs-data commit that published the database; `sha256` and `size` identify the object itself. Git LFS object ids *are* content sha256s, so this is the same identity the data repository records, and a lock can never point at an object sggs-data did not publish.

What the fetcher guarantees

scripts/data/fetch_dataset.py · `_stream_verified` · lines 132–154

```
def _stream_verified(lock: dict, dest: Path) -> None:
    """Download into a same-directory temp file, verify, fsync, then atomically replace
    dest."""
    db = lock["database"]
    href, headers = _download_href(lock)
    dest.parent.mkdir(parents=True, exist_ok=True)
    fd, tmp = tempfile.mkstemp(prefix=f"{dest.name}.", suffix=".part", dir=dest.parent)
    try:
        h, n = hashlib.sha256(), 0
        with os.fdopen(fd, "wb") as out, _open(_request(href, headers=headers), timeout=300)
as r:
            while chunk := r.read(CHUNK):
                h.update(chunk)
                n += len(chunk)
                if n > db["size"]:
                    raise SystemExit(f"download exceeded the pinned size ({db['size']}
bytes)")
                out.write(chunk)
            out.flush()
            os.fsync(out.fileno())
            if (h.hexdigest(), n) != (db["sha256"], db["size"]):
                raise SystemExit(f"download does not match the pin: sha256 {h.hexdigest()} size
{n}")
            os.replace(tmp, dest)
    finally:
        if os.path.exists(tmp):
            os.unlink(tmp)
```

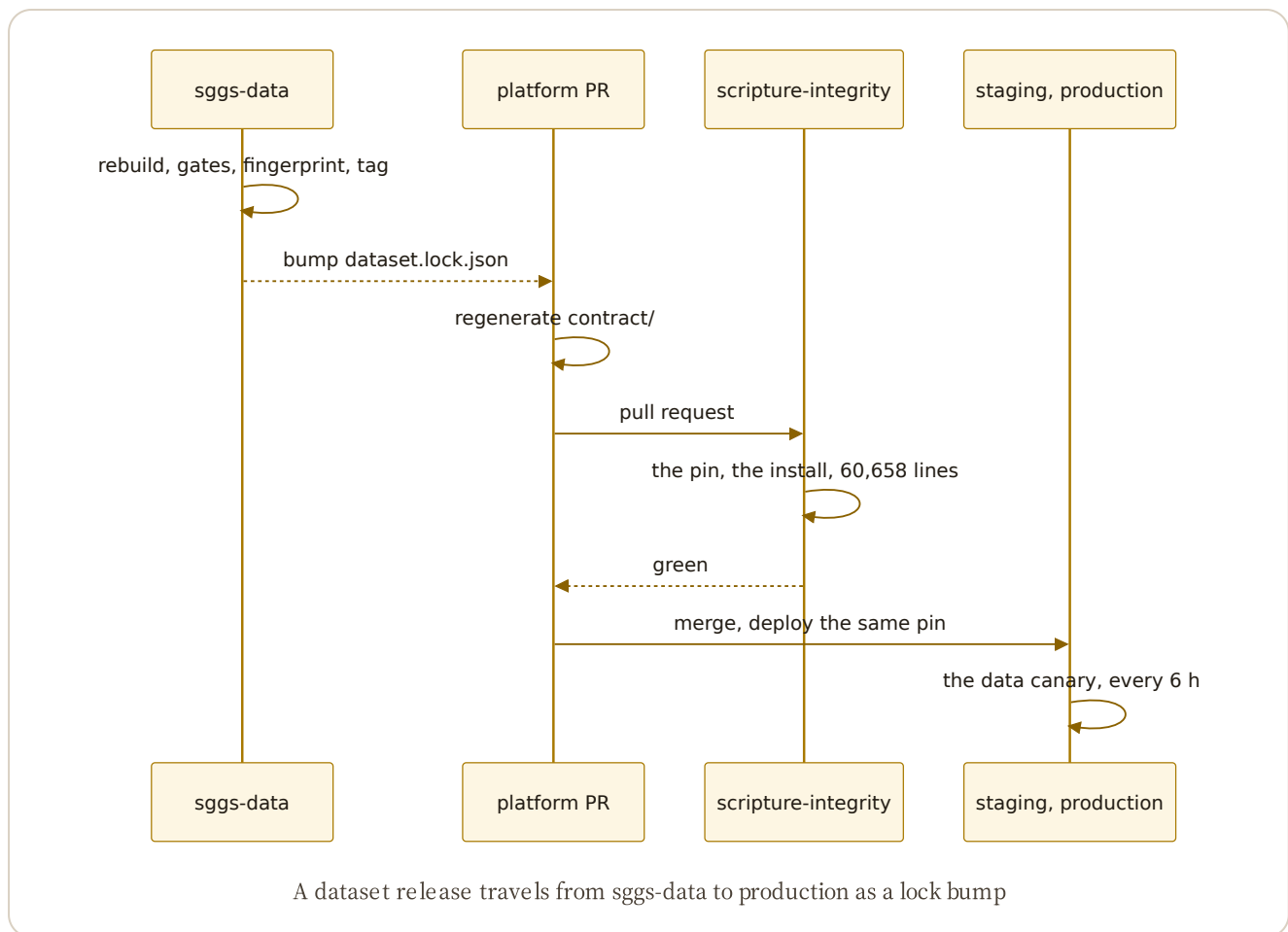
Read from the source at build time — [view these lines on GitHub at c3d9360](#) .

Source: [scripts/data/fetch_dataset.py](#)⁴, `_stream_verified`.

- The bytes written equal the lock's sha256 and size, or nothing is installed.
- The destination is replaced atomically (same-directory temp file, fsync, `os.replace`): a failed or interrupted fetch never leaves a truncated database behind.
- `--check-pin` reads the LFS pointer committed at the pinned sggs-data commit and requires it to name the same oid and size.
- `--check-repo` requires `contract/_meta.json` — the golden contract's own record of the database it was generated from — to name the same sha256, so a contract regenerated against a different database cannot ship silently.
- A token (`GH_TOKEN`) is sent when present; that is what a private sggs-data needs.

```
make dataset          # install db/sggs.sqlite from the pin (cached by hash in .dataset-
cache)
make dataset-check    # the pin agrees with contract/_meta.json, and sggs-data@commit
publishes it
```

How a new dataset arrives



1. sggs-data rebuilds, passes its gates, writes the fingerprint and `DATASET_VERSION`, and publishes the database by commit.
2. A pull request here changes `dataset.lock.json` and regenerates what is derived from the database: the golden contract (`make contract`), so `contract/_meta.json` names the new object, and the data dictionary (python3 `tools/gen_data_dictionary.py`). The docs check fails until both match the new pin.
3. The `scripture-integrity` workflow proves the pin and the contract agree, that sggs-data publishes the object at that commit, that the installed database opens, passes SQLite's integrity check and holds 60,658 lines over Angs 1–1430.
4. After merge, every deploy installs the same pin; the data canary samples lines from production every six hours and compares them byte for byte with the pinned database.

A revert is a lock revert: every pinned object stays available in sggs-data, so rolling back the data is the same small pull request in the other direction.

Which version is which

Three version numbers meet here, and they answer different questions:

Number	Where it lives	What it names	Today
APP_VERSION	webapp/serve.py; /api/health → version	the platform release: code, website, API	the release being served
dataset_version	dataset.lock.json	the sggs-data release the pin points at (DATASET_VERSION there)	1.0.0
meta.version	inside the database; /api/health and /api/meta → db_version	the build that produced the file	1.1.4

dataset_version and db_version differ today only because the pinned database was built on 2026-09-18, before sggs-data began stamping DATASET_VERSION into meta.version; the next rebuild makes them one number (**delivery**). The object itself is identified by its SHA-256, not by either version: to know exactly what a deployment serves, compare the lock's sha256 with contract/_meta.json and read db_version and commit from /api/health. The data dictionary lists every table of the pinned object.

The iOS app uses the same pin

gurbani-soul-ios carries its own copy of dataset.lock.json pointing at the same sggs-data commit, builds its bundled database pair from it (build_ios_db.py) and proves the database hash inside the archived app. Web, API and app therefore serve one dataset — see three repositories and pins.

Links

1. <https://github.com/Algorythmos-AI/sggs-platform/blob/c3d936031f0c283283c5d1102705affe1f8cf404/dataset.lock.json#L1-L10>
2. <https://github.com/Algorythmos-AI/sggs-platform/blob/integration/dataset.lock.json>
3. https://github.com/Algorythmos-AI/sggs-platform/blob/c3d936031f0c283283c5d1102705affe1f8cf404/scripts/data/fetch_dataset.py#L132-L154
4. https://github.com/Algorythmos-AI/sggs-platform/blob/integration/scripts/data/fetch_dataset.py

Sri Guru Granth Sahib Ji — Knowledge Base wiki. Generated from commit c3d9360. Updated 27 September 2026. Verified against aced1a0 on 2026-09-27. This copy may be out of date; the page is at <https://docs.gurbanisoul.com/data/dataset-pin/>