00 THE COMPLETE DOCUMENTATION

trentpower.fr

A signed, static, source-verifiable publication, explained end to end.

One reading of the whole system: what it is, how words become pages, how it is built, gated, signed and checked, how it is operated, and how it is kept private and ready for public scrutiny. Written so a curious non-technical reader and the engineer who maintains it can both follow it.

EDITION · 2026-06-24
FORMAT · A4 PORTRAIT · 210 × 297 MM
COMPOSED IN PARIS

PRIVATE · STATIC · SIGNED

· CONTENTS

What is inside

Each part opens in plain English and deepens as it goes. A reader in a hurry can stop after any opening summary; the detail and code are there when they are wanted.

A ORIENTATION

How to read this

A companion to the website, by the same author. Every part is built to be entered from the top and left whenever you have read enough.

A.1

THE RHYTHM

Plain words first, detail underneath

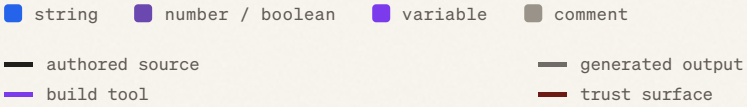
Every section follows one shape: a **plain-English summary**, then a **small picture** of how the thing works, then the **practical detail**, and finally a single line on **why it matters**. A business reader can stop after the summary and lose nothing essential. A developer reads on for the names, paths and commands.

A.2

READING THE MARKS

Three small legends

Diagrams read in grayscale; value and weight carry meaning, never colour alone. A single oxblood accent marks the one node that matters most. Code is shown only where it is genuinely illustrative, with light token colour.



A.3

WHO IT IS FOR

Two readers at once. To a non-technical reader it answers “what is this site, and why is it built so unusually?” To an engineer it is a map of the codebase from the first authored word through to the last signed byte on the live host. Where the two diverge, the summary serves the first and the detail serves the second.

B THE IDEA

What trentpower.fr is

A small, finished personal website published in English and French: a folder of plain files anyone can check for tampering, that collects nothing and depends on nothing.

B.1

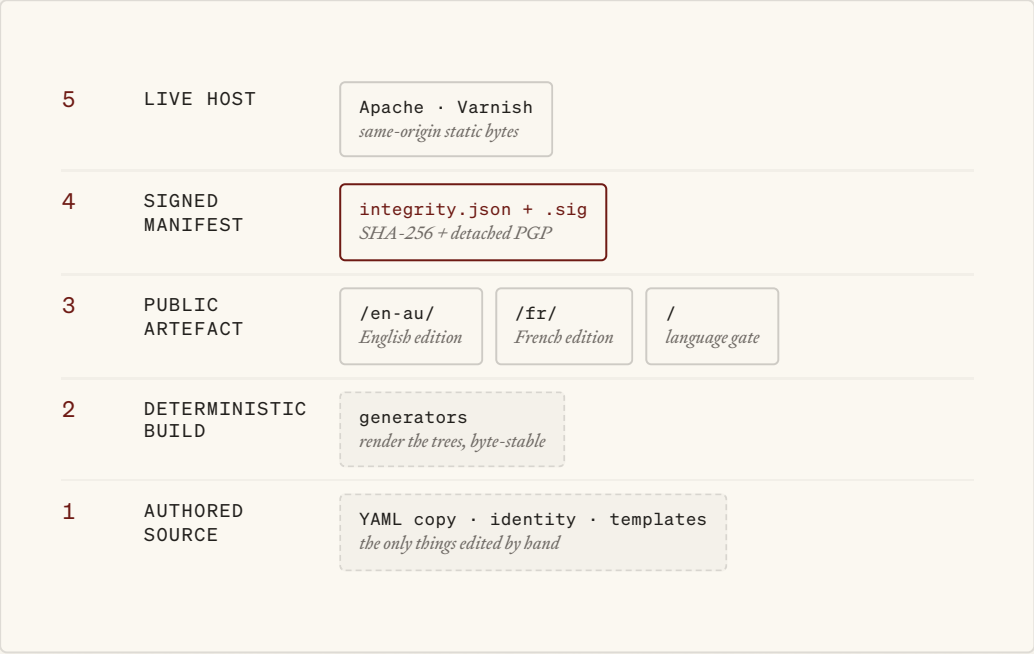
IN PLAIN TERMS

Most websites run software on a server every time you visit. This one does not. It is a directory of pre-made files (pages, images, fonts) served exactly as written. Each file is fingerprinted and the list of fingerprints is signed, so a stranger can confirm that what their browser received is what was published in the signed record. It is static, editorial, finite, and cryptographically signed.

B.2

THE SHAPE

authored input in, signed static bytes out; no request-time code



The whole system on one axis: authored source at the base, signed public bytes at the top. Nothing runs at request time.

B.3

THE DETAIL

Two complete editions ship: `/en-au/` (English) and `/fr/` (French, with localised slugs `confidentialite`, `securite`, `integrite`, `verifier`). The root `/` is a lightweight language gate carrying no editorial content of its own.

There is no runtime CMS, no database, no analytics, no cookies, no advertising identifiers, and no third-party runtime assets: fonts are self-hosted woff2 and every asset is same-origin. The browser storage that is used stays on the device: a short, fixed set of same-origin keys holding visitor preferences and first-visit markers, plus the service-worker offline cache. The storage itself is never sent anywhere, and all of it is listed and clearable on `/local/`. JavaScript is enhancement only (language switch, scroll reveal, project modal, citation overlay, service-worker registration); the page means the same with scripts disabled.

The build is deterministic on the author's machine: two consecutive local runs produce byte-identical output, modulo the rotating PGP signature salt. Independent off-machine reproduction is the stated next step, not yet a claim. No language model, AI service, or external API sits anywhere in the release path. And `public/` is intentionally tracked in git: the deployed bytes *are* what the signed manifest attests to.

0	0	0
COOKIES	TRACKERS	THIRD-PARTY REQUESTS
none set, ever	*no analytics, no pixels*	*all assets same-origin*

Measured at the network layer on page load. Source · the site's own CSP and privacy posture, as of edition 2026-06-24.

— WHY THIS MATTERS

A directory of static files has almost nothing to attack and almost nothing to maintain: no server code runs, no database can be queried, and every file a visitor receives is listed in the signed integrity.json manifest.

C THE CONTENT MODEL

From words to pages

All the site's words live in structured text, kept apart from the HTML. Writers edit prose in one place; the build sweeps it into every page, footer and citation.

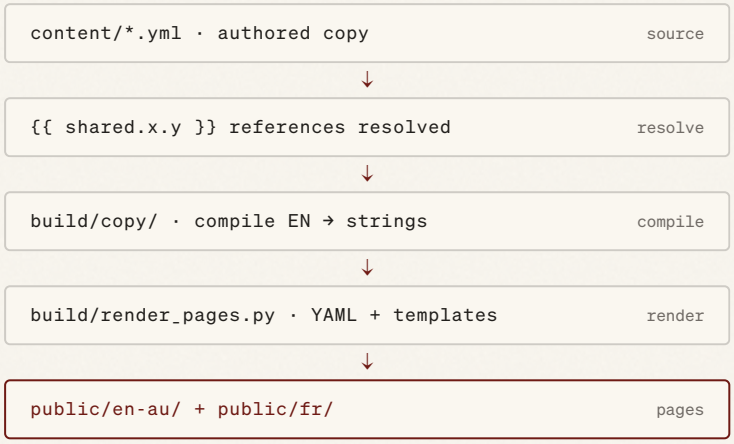
C.1

IN PLAIN TERMS

English is the authored record. French is curated by hand, because translation is editorial work. A phrase that appears in many places has exactly one source; change it once and it changes everywhere, in both languages, with no manual synchronisation.

C.2

THE PATH OF A WORD



C.3

THE DETAIL

Copy lives as YAML under `content/` : `en/shared.yml` plus `en/pages/*.yml` (home, privacy, integrity, source, verify, security, system, local) and the `fr/` equivalents. The route registry `content/routes.json` is generated from `content/shared/routes.yml` : 20 routes, one per logical page in each language, each carrying its path, locale, template, schema and trust flags.
The reference engine is deliberately dumb: one dotted path resolves to one shared string, with no conditionals and no loops. An unresolved `{{ }}` or a reference to a missing key is a **build error**; a long string repeated more than three times outside `shared.yml` is a **warning**.

CONTENT/SHARED.YML · SINGLE SUBSTITUTION — YAML

```yaml
# authored once, swept into every surface that quotes it
proof_line: "Every byte is hashed, signed, and checkable."

# referenced from any page, in either language:
tagline: "{{ shared.site.proof_line }}"
```

Public JavaScript is generated from `templates/*.template.js` ; never edit the generated `*.js` . The same single-source discipline governs identity: edition date, name and `sameAs` links all flow from `tools/config/identity_canonical.json` .

— WHY THIS MATTERS

One source per phrase means the two editions never drift, and a writer never hunts through HTML to fix a sentence.

D THE BUILD PIPELINE

One command, ten stages

A single command turns authored source into the finished, signed website. The order is fixed because some stages rewrite the HTML, and the fingerprints must be taken last, over the final bytes.

D.1

IN PLAIN TERMS

You run one command. It renders both editions, sweeps in the edition date and asset versions, mirrors the source, takes the cryptographic hashes, signs them, packages a frozen archive of the edition, and finally runs the safety gate that decides whether the result may ship.

D.2

THE TEN STAGES

order is load-bearing; hashing is always last

0	PREPARE	`build_copy · render_pages · hygiene` *compile EN, render trees, forbidden-artefact gate*
1·2	EMIT	`generate_site · generate_sw` *sweep edition / version / CSP; precache fingerprint*
3·4	MAP	`integrity (prelim) · verification_map` *hash tree, per-route verify records*
5·6	FIX & MIRROR	`generate_sri · source_view` *SHA-384 onto link/script; byte-equal /source/*
7·8	SEAL	`integrity (final) · gpg sign` *re-hash stable tree → integrity.json.sig*
9·10	PACKAGE & GATE	`release_archives · gate.py` *signed ZIP/TAR.GZ; blocking deploy gate*

Stages `generate_site` and `generate_sri` mutate the HTML; the final hash must run after them, or the browser blocks a stale-SRI script and Verify breaks.

D.3

THE DETAIL

The entry point is `bash tools/build/build.sh`. Two everyday invocations are the operator's whole mental model:

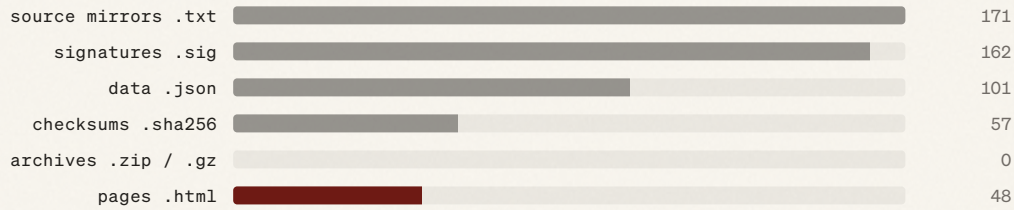
TOOLS/BUILD/BUILD.SH — SHELL

```
$ bash tools/build/build.sh           # full: generate, sign, archive, gate
$ bash tools/build/build.sh --check   # generators + gate only; no signing
```

Build and deploy are deliberately asymmetric: the **local build signs**; CI does not and never holds the PGP key. The runner only re-verifies the committed signature and uploads bytes. Deploy is a **non-deleting** two-pass SFTP mirror, so an accidental `git rm` cannot wipe live files. The CI deploy fires on manual dispatch only, with an explicit `confirm=DEPLOY`, so a fork's pull request can never trigger it. A post-deploy smoke test probes the expected-200 routes, the expected-403/404 denials, and re-verifies the live signature.

· WHAT THE PUBLIC FOLDER IS MADE OF

Of 636 published files, most are not "pages" at all. They are the verification scaffolding that makes the site checkable.



Tracked files under `public/`. 363 of them belong to frozen release archives. Source · `git ls-files public/`, edition 2026-06-24.

— WHY THIS MATTERS

The fixed stage order is what lets two runs of the same source produce the same hashes. Hash before the HTML is rewritten and the manifest describes bytes the browser never receives.

E GATES, CHECKS & QUALITY

What may ship

Before anything is published, the build runs a battery of automated checks. The serious ones block the deploy outright. A second tier of polish checks reports problems but never blocks an urgent fix.

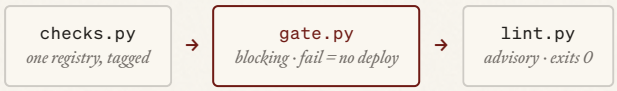
E.1

IN PLAIN TERMS

Is it signed? Does every file match its hash? Did any secret or local path leak? Those questions *block* a deploy if answered wrongly. Softer questions (search-engine hints, comment style, editorial copy) only *warn*, so a polish nit can never hold a security fix hostage.

E.2

TWO TIERS, ONE REGISTRY

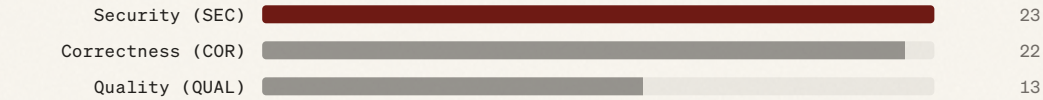


E.3

THE DETAIL

A single registry drives both runners. Every check carries a tier, a category, and a one-line rationale, in gate order.

58	45	13
CHECKS TOTAL	BLOCKING	ADVISORY
one registry, two runners	*fail = deploy stops*	*warns, never blocks*



By category. Source · `tools/lib/checks.py` *registry, edition 2026-06-24.*

CHECK	CAT	TIER	PROVES
gpg	SEC	block	signature verifies against the published key in a clean keyring
source_mirrors	SEC	block	every /source/ mirror byte-matches its live file
frozen_archives_immutable	SEC	block	sealed releases byte-identical to baseline
local_path_leakage	SEC	block	no /home/, Desktop/ or htdocs path in any shipped byte
bilingual_html / lang_gate	COR	block	both editions and the gate render correctly
lighthouse_invariants	QUAL	advise	accessibility / SEO budgets hold

A representative slice. Inline cross-cutting checks are functions in `tools/quality/inline_checks.py` *; the registry in* `tools/lib/checks.py` *feeds both tiers.*

Three more gates run in continuous integration, beside the registry rather than inside it. A full-history **secret scan** walks every commit for keys and local paths. A **dependency scan** (osv-scanner) checks the build toolchain against the OSV database and fails closed; a finding that cannot affect a static site with no runtime is recorded, with its reason, in a machine-readable VEX at `security/openvex.json`, so nothing is ever suppressed silently. A **licensing check** (`reuse lint`) proves every tracked file resolves to a licence and copyright. All three fail the pull request, not merely warn.

— WHY THIS MATTERS

Splitting the checks by severity means a missing signature or a leaked path stops the deploy, while a comment-style nit cannot. The two failure modes never trade places under pressure.

F TRUST & VERIFICATION

Proof anyone can re-run

Every public byte is fingerprinted; the list of fingerprints is signed with a key only the author holds. Anyone can confirm the site they received is exactly what was signed, without trusting the host or the network.

F.1

IN PLAIN TERMS

Re-download the live site and check it against the signed manifest using standard tools. Either it matches and gpg prints “Good signature”, or it does not and you should not trust the bytes. Past editions are frozen, so the historical record cannot be quietly rewritten.

F.2

THE CHAIN

runs in a throwaway keyring, leaves no residue

F.3

THE DETAIL

`integrity.json` maps every publishable path to a `sha256-…` digest; `integrity.json.sig` is a detached PGP signature over it. The public key sits at `/.well-known/pgp-key.asc`, fingerprint `A729 591B 450D 3F59 3694 98BD 8299 1F25 04AE 0263`. The whole check, made executable:

VERIFY THE LIVE SITE IN A CLEAN KEYRING — BASH

```
tmpdir="$(mktemp -d)"; export GNUPGHOME="$tmpdir"
ts=$(date +%s)
curl -fsS "https://trentpower.fr/.well-known/pgp-key.asc?ts=$ts" | gpg --import
curl -fsS "https://trentpower.fr/integrity.json?ts=$ts"      -o integrity.json
curl -fsS "https://trentpower.fr/integrity.json.sig?ts=$ts" -o integrity.json.sig
gpg --verify integrity.json.sig integrity.json
unset GNUPGHOME; rm -rf "$tmpdir" integrity.json integrity.json.sig
```

Beyond the manifest: `/source/` holds byte-equal text mirrors of every served file, so the bytes you read are the bytes that were signed. Frozen archives at `/integrity/releases/<date>/` carry signed `SHA256SUMS`, a `release.json` trust anchor, a dependency-free `verify.sh`, a `builds.json` index that records the canonical build and any same-edition rebuilds, and an `EXCLUDED_FILES.json` that names every file deliberately left out of the archive (and why) so a verifier can tell intentional omission from tampering. Sealed editions are immutable; any later drift is recorded as a parallel rebuild, never a mutation.

— WHAT IT PROVES

The bytes you received match a manifest signed by the held key. A stranger can re-run the check and get the same answer.

WHAT IT DOES NOT PROVE

Key identity: that the key is whose you think. Verify the fingerprint out of band.
Content truth: integrity only, never that a statement on the page is correct.
Freshness: an old-but-validly-signed release still verifies; the `?ts=` cache-bust fetches the current one.
Host & key safety: no defence against host or signing-key compromise.
Your device: not a compromised client, extension, or stale browser cache.

G SECURITY & PRIVACY

What the browser is told to enforce

The site collects nothing and loads nothing from anyone else, and the Content Security Policy tells the browser to enforce exactly that. Only an explicit allow-list of files is ever served; everything else is denied by default.

G.1

IN PLAIN TERMS

"We don't track you" is easy to say. Here the Content Security Policy tells the browser to block third-party connections and outbound requests, and the server serves only a named list of files; what little the site stores stays on your device, same-origin and visitor-controlled. On browsers that honour the policy, the claim is checkable rather than taken on faith.

G.2

DEFENCE IN DEPTH

4	MACHINE-CHECKABLE	`attestations.json · security.txt` *posture stated for machines*
3	ALLOW-LIST GATE	`.htaccess → [F,L]` *deny everything not allowed*
2	BROWSER POLICY	`CSP · COOP/COEP · HSTS` *default-src 'none'*
1	PRIVACY POSTURE	`no analytics / cookies / tracking` *nothing to leak*

G.3

THE DETAIL

The Content-Security-Policy is locked to `default-src 'none'` with `connect-src 'self'` on normal pages, so a page can talk only to its own origin and a fetch to any third party is not even possible. Scripts are `'self'` plus a small fixed set of `sha256` hashes; there is no `unsafe-inline`, no `eval()`, and no inline event handlers. Where the browser supports it, Trusted Types are required for any script sink (older browsers ignore the directive and fall back to the other `script-src` controls).

CONTENT-SECURITY-POLICY · .HTACCESS — HTTP

```
Content-Security-Policy: default-src 'none'; upgrade-insecure-requests;
  script-src 'self' <4 sha256 hashes>; script-src-attr 'none';
  style-src 'self'; font-src 'self'; img-src 'self';
  connect-src 'self'; worker-src 'self'; manifest-src 'self';
  frame-ancestors 'none'; object-src 'none'; base-uri 'self';
  form-action 'none'; require-trusted-types-for 'script'; trusted-types tp-app
```

Cross-origin isolation is retained (COOP `same-origin`, COEP `require-corp`), with HSTS at one year. Exposure is allow-listed rather than deny-listed: `tools/config/public-exposure.json` (kept under `tools/`, never deployed) is generated and then validated, and `public/.htaccess` runs a rewrite gate that ends in `RewriteRule . - [F,L]`, so anything not explicitly allowed is forbidden. The validator proves every reachable file is allowed, no file matches a deny rule, and every internal link resolves.

What the site stores stays on the device. A short, fixed set of same-origin keys holds visitor choices (appearance via `tp-theme`, last edition and reading position, first-visit markers), and the service worker keeps an offline cache named `tp-{edition}.{hashes}`, so a page read once works without the network. There are no cookies and no advertising identifiers; the storage itself is never sent anywhere, and `/local/` lists every key the visitor can inspect and clear.

Accessibility is part of the same posture: the site targets WCAG 2.2 AA. Each edition's signed `/tests/` snapshot records an automated accessibility check, and keyboard navigation, landmarks and focus order are verified by hand; known exceptions are stated on `/security/` rather than implied.

— WHY THIS MATTERS

Most privacy claims rest on a host's word. Here a visitor can read the response headers and the `.htaccess` gate and confirm the rules for themselves: `connect-src 'self'` means the page cannot reach a tracker even if one were added by mistake.

H OPERATIONS & RECOVERY

The deploy routine, and recovery

Running the site is a small, repeatable routine. Recovery from a stale cache, a bad signature, or a suspected compromise has a written procedure for each case.

H.1

IN PLAIN TERMS

Set one canonical date, rebuild so it propagates everywhere, gate, sign, deploy, then test the live site. Because the system is small, most problems are “restore the known-good copy and re-sign”, not improvisation under pressure.

H.2

EVERY DEPLOY

`edit source` *YAML / identity* → `build.sh` *propagate* → `gate.py` *must be green* → `sign` *detached PGP* → `deploy` *no --delete* → `smoke test` *verify live*

H.3

THE DETAIL

Filenames stay stable (`styles.css` , `sw-register.js`); the version rides in a query string instead, `styles.css?v=2026-06-13.4e425527` , where the value is the build's `asset_version` (the edition date plus a content hash). A changed build rotates that hash, so the browser fetches fresh bytes rather than a stale cached copy. The service-worker cache name encodes the same edition and content hashes, `tp-{edition}.{bundle_hash}-{precache_hash}-{release_tag}` , so any changed file rotates a hash and the old cache is purged on activate.

THE CRITICAL OPERATOR RULE

Never sign while `GNUPGHOME` points at the temporary verification keyring. If a signature fails for no clear reason: `unset GNUPGHOME` and restart the gpg agent, then sign again.

Incident response begins by freezing: reproduce from a private window, another device, and a mobile network *before* touching anything, since most “incidents” are a cache blip or a browser extension, not a compromise. If integrity genuinely fails, lock the host account, restore from the known-good local copy, regenerate, re-sign, and purge the cache. A temporary stale page or a Lighthouse wobble is explicitly *not* an incident.

THE BUILD CEREMONY AT ITS FIRST CHECKPOINT: NO EDITION IS CREATED OR REBUILT WITHOUT A DELIBERATE CHOICE

I THE SCORE LEDGER

A tool that never touches the site

One local-only tool sits beside the site but never touches what ships: it audits the live site after deploy, as evidence rather than as a gate.

I.1

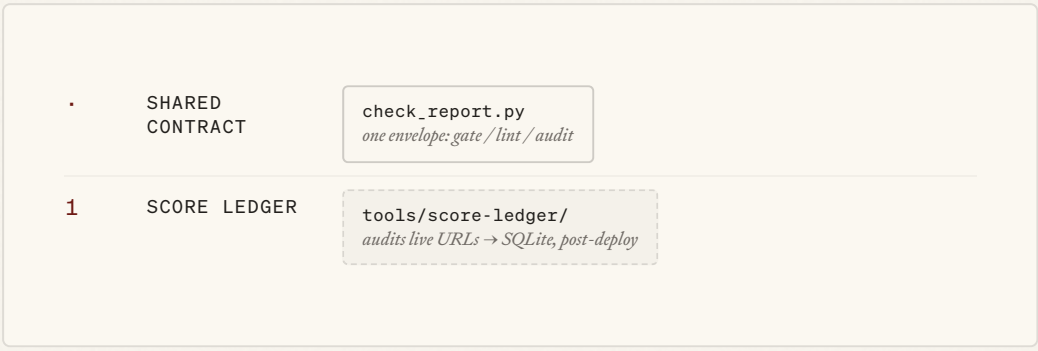
IN PLAIN TERMS

The *score ledger* watches the live site over time and records how it is doing, but nothing in the build ever asks its opinion. It shares one machine-readable report envelope with the gate and the lint, so every run of every tool reads the same way.

I.2

WALLED OFF

local-only; never touches public/



I.3

THE DETAIL

The score ledger is observational: it tests the **live** site over HTTPS, so you deploy and let it propagate first, then run it from the workstation to a local SQLite and reports. It stores atomic facts (run, target, tool result, metric, observation, evidence) and emits no single overall score. Lighthouse performance on a small machine is noise; the ledger trusts the rolling median of the last five runs, not any single number.

SHARED REPORT HEAD · CHECK_REPORT.PY — JSON

```
{ "schema_version": 1, "command": "gate", "status": "passed",
  "summary": { "passed": 35, "failed": 0, "warnings": 0 } }
```

J PUBLIC & AUTHORSHIP

Public by design, kept that way

The repository has been public since the first signed edition. What keeps it safe is not a one-time sweep but a standing discipline: no key, secret or local path is ever committed, and how the site is authored is stated plainly in the open.

J.1

IN PLAIN TERMS

The source is public, so every private thing must stay absent: keys, passwords, server paths. The repository also says plainly how it is made: written and reviewed by a human, with selective AI help for drafting only, and never auto-published. That is recorded once in public files, not stamped on every commit, and re-checked on every change rather than waved through once.

J.2

THE SWEEP

J.3

THE DETAIL

Everything private is `.gitignore`-covered: SSH keys, exported private `.asc` keys, TOTP secrets, service-account JSON, the `private/` tree, local credentials and databases. The published key at `/.well-known/pgp-key.asc` is intentionally public; a private exported key must never enter history. The deploy gate already enforces the public-bytes subset, and the readiness pass covers the wider repository the gate does not scan.

Authorship is stated once, canonically, and made machine-readable:

ATTESTATIONS.JSON · MACHINE-READABLE AUTHORSHIP — JSON

```
"automated_publishing": false,
"manual_review_before_publication": true,
"language_model_assistance": "selective drafting or structuring only"
```

A build-time check (`tools/quality/validate_git_metadata.py`) scans the tracked tree for forbidden commit trailers (`Co-authored-by` , `Generated-by`) and fails the build on any regression, so the commit log stays an editorial record, not a tool-attribution diary.

K PROVENANCE & RELEASES

Built in the open, checkable to the commit

The author's signature proves who published a record. A second, independent proof now rides alongside it: each release is rebuilt on a public machine and carries a tamper-evident record of exactly which commit and which workflow produced it.

K.1

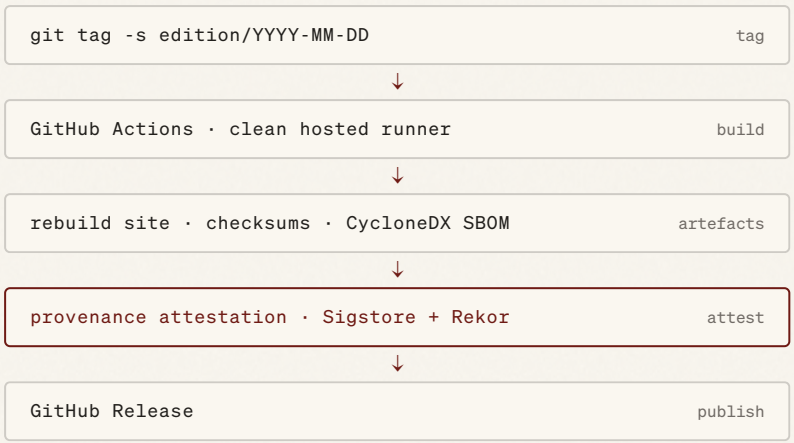
IN PLAIN TERMS

The PGP signature says “the author published this.” The new proof says “a clean build machine, which no one edits by hand, rebuilt this release from the tagged source, and here is the exact commit and workflow that did it.” That statement is written to a public, append-only transparency log, so anyone can check it and no one, the author included, can forge it after the fact. The two proofs are complementary: identity from the key, build history from the platform.

K.2

THE RELEASE PATH

a signed tag starts it; the runner does the rest



K.3

THE DETAIL

Pushing a signed `edition/YYYY-MM-DD` tag triggers a release workflow on a GitHub-hosted runner. It re-renders the site from the tagged source (a reproducibility proof that fails closed if the committed bytes do not rebuild), packages a deterministic archive, and asks the platform to attest it. The attestation is signed **keyless** through GitHub's OpenID Connect identity and Sigstore, and recorded in the public **Rekor** transparency log. This is SLSA build-track **Level 3**: the build runs on a hosted service and the provenance cannot be falsified by the build steps themselves. The canonical record stays locally built and PGP-signed; the signing key never enters the runner.

VERIFY THE BUILD PROVENANCE OF A RELEASE ARCHIVE — BASH

```
$ gh attestation verify trentpower-fr-2026-06-13-site.tar.gz # commit + workflow
    --repo trentpower/trentpower.fr
```

Each release also ships a **CycloneDX SBOM**. The site itself has no runtime dependencies, so the bill of materials records the build toolchain rather than anything shipped to a visitor. Archives are **byte-deterministic** (sorted entries, pinned timestamps, no host metadata); rebuilding from the same source yields identical bytes, with the deliberate exceptions of the random signature salt and the licensed fonts. The full reproducibility story is in `docs/REPRODUCIBILITY.md`.

Independently checkable standards. The same posture is measured by outside parties, not just asserted here: the OpenSSF **Best Practices** badge (passing, at the silver tier) and the OpenSSF **Baseline** criteria; **REUSE** 3.3 compliance, so every tracked file resolves to a licence and copyright by machine; and the OpenSSF **Scorecard**, treated as a maintenance dashboard rather than a medal, and capped by design for a single-maintainer project that cannot require a second reviewer.

— WHY THIS MATTERS

A signature alone asks you to trust one machine. Provenance built on a public runner and logged to Rekor lets a stranger confirm where a release came from without trusting the author's laptop, the host, or even GitHub's word after the fact.

K.4

CONTENT CREDENTIALS

provenance that travels with the file

The proofs above secure the *publication*. **C2PA Content Credentials** add a portable, file-level layer for the few assets that travel on their own: the authored architecture diagrams. Each is published as a signed copy under `/provenance/` carrying an embedded credential: who published it, its canonical URL, its AI posture, and a pointer back to the signed integrity manifest. The credential is **self-signed**, so it binds a consistent claimed identity, not a trust-listed one, and a public verifier shows the signer as "untrusted." The PGP key stays the identity anchor, with the signing certificate's fingerprint published beside it; the signed bytes are verified by hash through `integrity.json`, not byte-reproduced. `README.pdf` itself cannot carry an embedded credential, because the C2PA toolchain has no PDF writer, so the document is proven at the site level only. The policy and scope are in `docs/C2PA.md`.

L APPENDIX

The map, and the principles

The whole repository at a glance, authored source through to signed public bytes, and the order in which its principles win when they conflict.

L.1

REPOSITORY MAP

colour marks each file's role

```
trentpower.fr/
├─ content/            authored copy (YAML) + route registry
│   ├─ shared/routes.yml  en/  fr/
├─ templates/          *.template.js → generated JS
├─ styles/             authored design source CSS (*.src.css)
├─ tools/              the pipeline, one dir per responsibility
│   ├─ build/           creates the site (build.sh, generators, copy/)
│   ├─ quality/         stops a bad release (gate.py, lint.py, validate_*)
│   ├─ verify/          proves the release is genuine (read-only)
│   ├─ release/         makes it public (archives, seal, deploy.sh)
│   ├─ config/          declared facts (identity, public-exposure)
│   ├─ lib/             shared across pillars (paths.py, checks.py)
│   └─ score-ledger/    local live-site audit
├─ public/             SIGNED, DEPLOYED bytes; the web root
│   ├─ en-au/  fr/  index.html  editions + language gate
│   ├─ integrity.json (+ .sig)
│   ├─ .well-known/pgp-key.asc
│   ├─ source/          byte-equal mirrors
│   └─ integrity/releases/<date>/  frozen archives
├─ docs/               this documentation set
└─ README.md  README.pdf
```

authored source
build tool
generated output
trust surface

L.2

PRINCIPLE ORDERING

Every choice in this repository answers to a fixed ordering of principles. When two of them pull in different directions, the one earlier in the list wins:

PRINCIPLE ORDERING — TEXT

```
1 Privacy → 2 Security → 3 Verifiability → 4 Accessibility →
5 Minimal surface → 6 Determinism → 7 Human-readable → 8 Finite
```

· COLOPHON

Colophon

· TYPE

Set in Klim Type Foundry's **Signifier** (serif, the reading and display voice, Light 300 throughout), **Söhne** (sans, captions and support only), and **Söhne Mono** (paths, hashes, labels, numerals): the same families the live site serves. If the families cannot be embedded, substitute Source Serif · Inter · JetBrains Mono, which keeps roughly seventy per cent of the voice.

· COLOUR

Warm paper, iron ink, a single oxblood accent. No semantic colour layer; state is carried by type, rules and language. Code tokens are the only place jewel-tone pigment appears, and every diagram reads in grayscale.

· HOW THIS WAS MADE

Composed in a single authored HTML document (`docs/pdf/readme.html`) using the project's print style system, paginated to A4 with **paged.js**, and exported through headless Chromium. The HTML is the source of truth; this PDF is its export. When the system changes, the HTML is updated and the PDF re-exported, never hand-patched. After every export the rendered pages are checked for clipped text, overlapping blocks, missing assets and chart proportions before the document ships.

— PROVENANCE

Edition 2026-06-24 · A4 portrait · composed in Paris. Charts and figures are generated from repository data (`tools/lib/checks.py`, `git ls-files public/`) and validated against it at export time, not asserted. The operator-console capture is a tracked documentation asset, taken from the live build ceremony and set in the publication's own mono face.

trentpower.fr · Private · Static · Signed