

Ship & operate

CI Gates — what each check proves

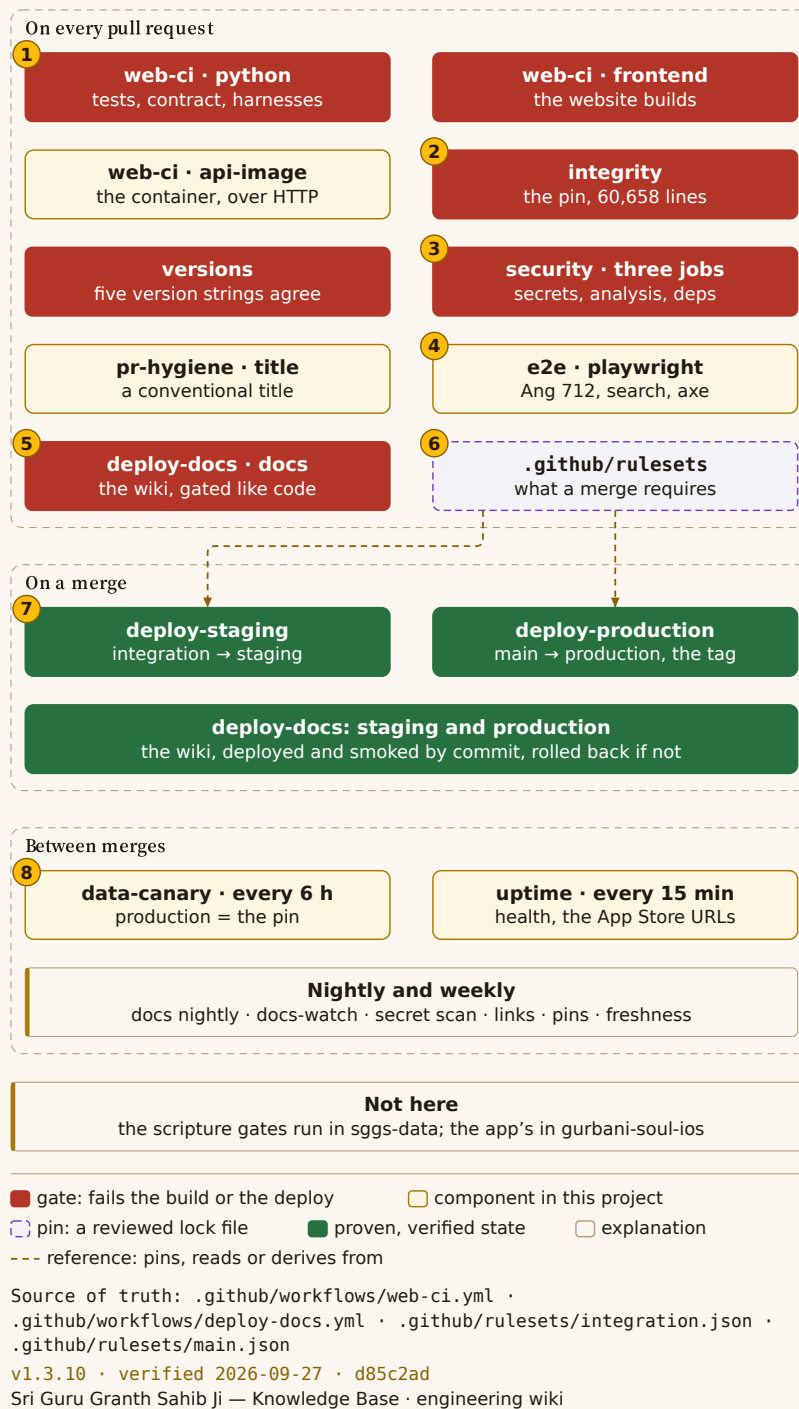
🕒 7 min read · Updated 10 October 2026 · Verified against 1560d98 on 2026-09-26

What each GitHub Actions workflow and job proves before a change can merge or deploy.

<https://docs.gurbanisoul.com/process/ci-gates/>

10 • The CI gates map

Every GitHub Actions workflow and job — what it proves, when it runs, and which checks a branch ruleset requires before a merge



Poster 10 — The CI gates map: every workflow and job on a pull request, what a ruleset requires, the deploy chains on a merge, and the schedules between merges

- web-ci: the code.** python runs ruff, the server and repo-gate unit tests, replays the golden contract over HTTP, checks that contract/openapi.json is current and covers every dispatcher route, that /api/health is all-true, the shabad-heading regression and the search harnesses. frontend builds the Astro site and passes the pahar vectors. api-image builds the Docker image and proves the running container reports the commit and honours the contract.

2. **The data pin and the version.** integrity proves dataset.lock.json and contract/_meta.json name the same object, that sggs-data publishes it at the pinned commit, and that the installed database opens, passes quick_check and holds 60,658 lines over Angs 1–1430. versions proves the five platform version strings agree and that a main PR comes from integration or a hotfix branch.
3. **Security and hygiene.** gitleaks (secrets), bandit and semgrep (static analysis), actionlint, and a blocking npm audit at level high on frontend/ and docs-site/ (deps). pr-hygiene requires a conventional-commit title.
4. **End to end.** The Playwright suite drives the real site against serve.py and the pinned database: the Ang 712 heading, search to the panel heading, and axe on the home page. It is not a required check, but both deploy chains wait for it on the exact commit.
5. **The docs job.** The wiki is gated like the product: the sibling pin is consistent and installed; tools/docs_check.py checks frontmatter, links and anchors, widget fallbacks, the Mermaid palette, the scripture-quotation rule, the posters, the verified stamps and drift against the code; the generated routes page is current; the plugin and tool unit tests pass; the types check; the site builds with every Mermaid fence rendered; the built site is checked whole — every page rendered, every internal link resolving, the page budgets, and every inline script allowed by its CSP hash; Playwright and axe run with a mocked API under the production headers; Lighthouse passes its budgets.
6. **What a ruleset requires.** The JSON files under .github/rulesets are the source of truth, applied by scripts/gh/apply_rulesets.sh. integration requires python, frontend, integrity, secrets and docs; main requires python, frontend, integrity, versions, secrets, static-analysis and docs with the strict policy, plus a review and CODEOWNERS. Neither allows deletion or force-push; main has no linear-history rule because release PRs are merge commits.
7. **On a merge: the deploy chains.** A merge into integration runs deploy-staging: gates on the commit, preflight for secrets, the API functions generated and proven, the web built and aliased, then verify (every function ready at the commit, the gateway, the smoke, the whole contract). A merge into main runs deploy-production: every required check except the wiki's docs job (which gates the merge, not the product) plus playwright, preflight, an optional approval, API deploy and verification by commit, web built unaliased and smoked, promotion with the public smoke and a rollback edge, then the tag. The wiki has the same chain in deploy-docs: its staging deploy refuses to be the project's first deployment (Vercel made that one production), its rollback targets the deployment the domain actually serves, and every production failure — or a docs job that failed on main — opens an issue.
8. **Between merges.** Every six hours the data canary proves production serves the pinned scripture byte for byte — 500 sampled lines and fifteen Angs, at the API origin and through the site's CDN — and replays the golden contract; every fifteen minutes uptime probes /api/health and the App Store URLs; every fifteen minutes uptime also probes the wiki; every six hours docs-watch proves docs.gurbanisoul.com serves the commit its last production deploy shipped and runs the live suite against the real API; every night the whole docs job re-runs on integration to catch drift from outside the repository; on Mondays security scans the full history for secrets, docs-links checks every external link in the wiki and the sibling docs it publishes, docs-pins moves a sibling pin when the docs it publishes changed, and docs-freshness lists the pages whose cited code moved since their verified stamp. Each opens or updates one issue; none pages or redeploys.

Workflow · job	Proves	Needs
web-ci · python	ruff clean; server+ repo-gate unit tests; the golden contract replayed over HTTP (tools/contract_http.py); contract/openapi.json is current and covers every dispatcher route; /api/health all-true; the shabad-heading regression (api_superset_check); search harnesses; contract does not drift	pinned DB

Workflow · job	Proves	Needs
web-ci · frontend	Astro builds; pahar vectors pass; report-only : the committed webapp/static equals the build's text assets byte for byte (scripts/ci/check_static_fresh.py — a warning, not yet a gate)	Node 22
scripture-integrity	dataset pin: dataset.lock.json ↔ contract/_meta.json , and sggs-data publishes that object at the pinned commit; the installed DB passes quick_check with 60,658 lines over Angs 1–1430 (the scripture gates themselves run in sggs-data)	pinned DB
version-consistency	all 5 platform version strings unified (the dataset versions in sggs-data; the iOS app keeps its own match); main PRs come from integration/hotfix	—
security	gitleaks; bandit; semgrep OSS; actionlint; blocking npm audit --audit-level=high on frontend/ and docs-site/ (job deps)	—
pr-hygiene	conventional-commit PR title	—
e2e · playwright	Ang 712 heading smoke, search → panel heading, axe WCAG (serious and critical) on /, /search, /features, /watch and /learn ; the Knowledge Base shell on every KB route and the 404 page (frontend/e2e/kb-shell.spec.ts): unique ids, a viewport that still pinch-zooms, and axe in the light and the dark theme against a per-page list of known violations that can only shrink; sideways scroll on a 375 px phone is measured and reported as a warning — the Knowledge Base sets its Latin text in the system font, so that baseline is per platform, enforced on macOS today and on the Linux runners once a Linux baseline is recorded in the spec. One job, two steps: the desktop project (Chromium), then the phone project (iPhone 13, WebKit)	serve.py + pinned DB
e2e · kb-lighthouse (report-only, never blocks)	Lighthouse with mobile emulation on /search, /reader?ang=1 and /nitnem , served by serve.py ; three runs each, no assertions (frontend/.lighthouse.rc.json); the reports are the kb-lighthouse artifact	serve.py + pinned DB, Chrome
web-ci · api-image	Docker image builds; the running container reports the build commit and honours the golden contract over HTTP	Docker
deploy-verify (on Vercel deployment_status , previews only)	preview /api/health all-true (passes with a notice without the repo-level bypass secret)	—
release (manual fallback)	idempotent tag+ GitHub Release on main ; normal releases are cut by deploy-production after a verified deploy	—
data-canary (every 6 h, not a PR check)	production serves the pinned scripture byte for byte: 500 random lines (/api/lines) and 12 random Angs plus Angs 1, 712, 1430 (/api/ang/N), from the API origin and through the public site's CDN; the golden contract replays against production; a failure opens or updates one issue	pinned DB
deploy-staging · deploy-web	the API functions are generated at the commit from the pinned database (slices proven); after vercel build every function bundles only its entry, the API code and its own database, and no database or API source is in the static output	pinned DB
deploy-staging · verify	every API function's /readyz reports this commit and exactly its contexts; no API file is downloadable; the web smoke; every routed context answers through the gateway with its own X-Service ; an unknown prefix falls through to all ; the whole golden contract passes through the gateway	bypass secret, pinned DB

Workflow · job	Proves	Needs
deploy-docs · docs (required on integration and main)	the wiki (docs-site/ over docs/ , ADR-0012): pinned sibling docs consistent and installed; tools/docs_check.py (frontmatter, links and anchors, widget fallbacks, Mermaid palette, the scripture-quotation rule, posters, the verified stamps on process/engineering/architecture pages, drift against the code: search modes, verify thresholds, generated pages; the poster legend; the site theme's brand colours equal docs/brand/tokens.json ; the CSP allows inline scripts by hash only); tools/gen_route_table.py --check ; scripts/brand/contrast_report.py --check (the committed contrast report equals a regeneration from docs/brand/tokens.json , frontmatter included); tool and plugin unit tests; types (astro check); the site builds with every Mermaid fence rendered to SVG ; every page rendered completely; every internal link and fragment in the built HTML resolves; page budgets (JS ≤ 60 KB gz); every inline script allowed by its sha256 in the CSP, no inline handlers (scripts/csp.mjs); Playwright e2e (mocked API) + axe on every built page, served with the production headers and CSP (a blocked script fails the check); Lighthouse (performance ≥ 0.9, accessibility 1, best-practices ≥ 0.95) on five representative pages for a pull request or a push to integration , and on all twenty nightly and on main ; every page's Open Graph card exists	Node 22, Chromium
deploy-docs · deploy-staging/ deploy-production	the wiki is deployed only by CI: integration → sggs-docs-staging.vercel.app (refuses to be the project's first deployment, fails unless its own deployment is a preview); main → record the deployment the domain serves, build, deploy unaliased, smoke by commit, promote to docs.gurbanisoul.com , smoke again, roll back to the recorded deployment if the public smoke fails; every failure opens an issue (scripts/ci/docs_smoke.py , scripts/ci/vercel_api.py). The smoke fails on any redirect. After every staging deploy the live suite runs against staging with the bypass header	docs Vercel project secrets
deploy-docs · main-guard	a push to main whose docs job did not succeed never deploys the wiki; this job opens an issue saying so	—
deploy-docs nightly (schedule)	the whole docs job re-runs on integration every night, to catch drift from outside the repository (npm, Starlight, sibling docs, the dataset); a red night opens one issue, the next green night closes it	—
docs-links (weekly, not a PR check)	every external link in docs/ and in the sibling docs the wiki publishes still resolves (lychee, policy in .lychee.toml); a failure opens or updates one issue, never blocks	—
uptime (every 15 min, not a PR check)	production/api/health all-true; /privacy and /support (the App Store URLs) resolve with their content; a failure opens or updates one issue. Job docs : the wiki's landing page and /api/health through its rewrite, with no redirect (its own issue)	—
docs-watch (every 6 h, not a PR check)	docs.gurbanisoul.com serves the commit the last successful deploy-docs run on main shipped (a domain changed outside CI fails it), docs_smoke.py passes, and the live suite (docs-site/e2e-live) passes against the real API: status strip, search, the Ang explorer on Angs 1/712/1256, a line read from the API verifying exact, the API console, 𐆆K search, axe as deployed, no CSP errors. One issue, closed when green	—

Workflow · job	Proves	Needs
docs-pins (Mondays, not a PR check)	when a sibling repository changes a file the wiki publishes, a pull request moves its pin (<code>tools/docs_pins.py</code> ; a commit that touched nothing published moves nothing); opened with DOCS_BOT_TOKEN so the required checks run (after a dry-run push proves the token can write), or one issue — naming the cause — when the secret is absent, cannot push, or the pull-request step fails	DOCS_BOT_TOKEN (optional)
docs-freshness (Mondays, not a PR check)	lists the stamped pages whose cited code (excerpts, linked source files, the posters' "Source of truth") changed since their verified stamp, then pages far behind; one issue rewritten weekly, closed when every page is fresh (<code>tools/docs_check.py --freshness</code>)	—

The PDF, the corpus rebuild and the scripture gates (reconcile attestation, regroup invariants, scripture guard, editorial ledger, fingerprints) belong to **Algorythmos-AI/sggs-data** and run in its CI. The iOS app's gates run in **Algorythmos-AI/gurbani-soul-ios**.

Run the no-PDF subset locally with `make ci`.

`npm audit` has been **blocking** since the Astro 7 upgrade cleared the advisory backlog (2026-09-20): a new high/critical advisory in `frontend/package-lock.json` fails `security · deps`. Fix it with `npm audit fix` (never `--force` blind — read the migration notes), rebuild, and re-sync `webapp/static/`; do not re-add `|| true`.

Deploy pipeline

`deploy-production` (push to main) waits for the required checks above **plus playwright** on the exact commit, then deploys API → verifies → builds web unaliased → smoke → promotes → smoke public → tags. Details: [runbook: deploy](#).

Not yet wired (roadmap)

`perf` (latency budget + Lighthouse on staging) and a load/soak test — nothing yet exercises the request-cost limits this repo now enforces (the `/api/verify` claim cap, the socket timeout, the bounded worker pool). `npm audit` is already blocking (see the `security` row above).