NLTlabs
NEW LOGIC TECH

NLT Quality Pipeline

Builder Ops · dev-time gates & docs

Quality MCP · docs MCP · gates · judges · skills.

44 curated MCP tools across 9 families in this volume.
Dev-time scoring and checklist — distinct from runtime memory (NLT Memory Deep Dive).
Bootstrap consumers with init; add validate_changed judges in document repos.
Repo paths and CLI names stay internal; the outward suite uses NLT component names.

AUDIENCE Agent authors · platform maintainers · consumer repo operators

Version · nlt-quality **Issued · 2026-09-25**
NLT Labs · New Logic Tech

Executive summary

44	9	5	Studio
MCP TOOLS	TOOL FAMILIES	PIPELINE STAGES	REPORT STUDIO
Curated surface	Scoring · docs · Linear	Discover → verify	Report Studio Builder Ops

NLT Quality Pipeline is the dev-time scoring and checklist layer for AI coding assistants, distinct from runtime NLT Memory Deep Dive. Consumer repos bootstrap via init; document repos add validate_changed judges for PDF and site output.

Key findings

1. The quality-gate server exposes deterministic gates, not LLM judgment at runtime.
2. docs-mcp validates Linear issue bodies before tracker writes.
3. Hooks and skills enforce finish-task before session end in consumer IDEs.

Who should read this
Agent authors and consumer-repo maintainers. Investors start at NLT Labs Portfolio Story; memory operators use NLT Memory Deep Dive and the NLT Brain UI guide on briefs.nltlabs.ai.

Contents

How to read this. This is the maintainer runbook for the NLT Quality Pipeline MCP servers — the quality-gate server and docs-mcp, built from one monorepo checkout (a sibling of this repo by default). Read Parts I–III to place the pipeline, Parts IV–VI to configure tools and judges, Parts VII–IX to operate day to day. For memory tiers, bridge flow, and hive propagation see **NLT Memory Deep Dive**; for the report-studio CLI see **Report Studio Builder Ops**.

Part I — Pipeline role and boundaries

PART I

NLT Quality Pipeline at a glance

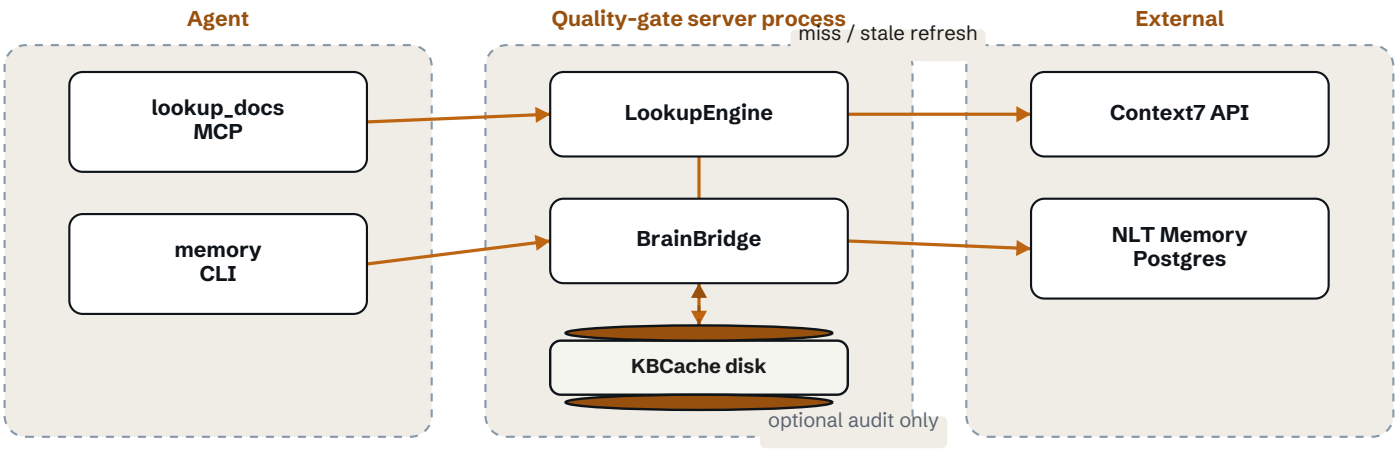
NLT Quality Pipeline is the dev-time quality and documentation layer for AI coding assistants. The quality-gate server exposes deterministic tools — scoring, security scans, quality gates, checklist enforcement, and Linear cache wrappers. docs-mcp generates README, changelog, and validated Linear issue bodies. Neither server embeds consumer logic: each consumer repo (ReportLab, NLTlabsPE, AgentForge) pins its own server version and configures judges in its project quality config for the output it ships.

Three-way distinction

Layer	Brief	Runtime?
NLT Memory (brain HTTP API)	NLT Memory Deep Dive	Yes — agents recall during work
Report Studio (PDF engine)	Report Studio Builder Ops	Build time — briefs.nltlabs.ai
NLT Quality Pipeline (this volume)	NLT Quality Pipeline, Builder Ops	Dev time — IDE MCP session only

QUALITY-GATE SERVER — LOOKUP AND MEMORY INTERACTION

Level 2 · MCP host calls cross process boundaries; brain is never in-process in consumers.



Solid = primary path · dashed = optional or audit-only.

Figure — LookupEngine serves docs cache-first; BrainBridge reaches memory over HTTP only. Call the MCP tools — never query the NLT Memory Postgres directly.

> "Do not conflate quality-pipeline session hooks with the federated NLT Memory service — reach NLT Memory over the HTTP bridge only, never as an in-process embed."
>
> **— Consumer boundary**

Part II — Monorepo package map

PART II

Packages and consumer wiring

Component	Role
Core package	Shared config, metrics, brain bridge, judges
Quality-gate server package	MCP server — scoring, gates, checklist, Linear wrappers
docs-mcp package	README/changelog/Linear doc generators (companion)
Combined platform entry	Serves both MCP servers as one process (platform CLI)
Project quality config	Consumer preset: engagement, judges, memory profile
Agent skills	Finish-task, handoff, validate bundles
CI quality workflow	PR gate on changed Python

Bootstrap path. Run init once per consumer repo: it writes AGENTS.md, TECH_STACK.md, platform hooks, skills, and optional .mcp.json bridge config. Pass with_report_studio=True to add the report-studio consumer scaffold. The generated MCP config strips any direct NLT Memory entry — memory goes through the HTTP bridge only.

Consumer wiring. ReportLab, AgentForge, and NLTWeb each reach the quality-gate server over the MCP bridge — never embed brain Python in a consumer tree. docs-mcp handles Linear issue templates and README generation. Set the quality preset in the project quality config; the engagement level decides which hooks are wired, from SessionStart through Stop.

> "Every package ships as an editable install from the monorepo — run uv sync at the workspace root to keep checkers aligned with CI."
>
> **— Monorepo discipline**

Part III — Pipeline stages

PART III

NLT quality pipeline

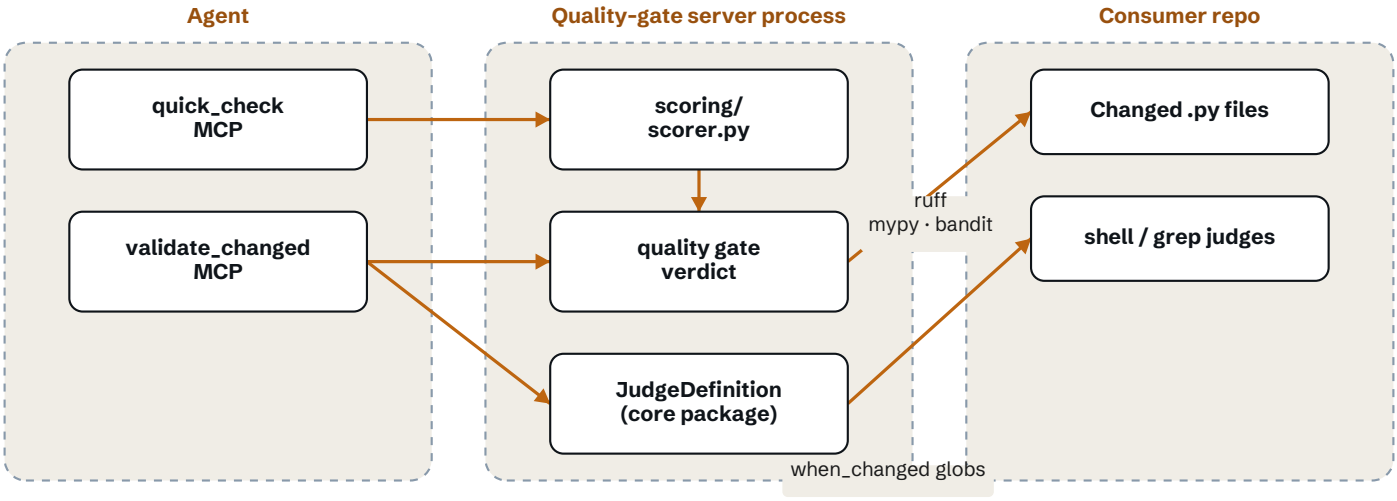
Run every session through five stages: discover, research, develop, validate, verify. Do not skip research — omitting the docs lookup (lookup_docs) is the most common source of hallucinated APIs. Do not skip verify — omitting the completion checklist (checklist) is the most common source of undeclared work.

Stage	Primary tools	Done when
discover	session_start, server_info	Checker env and brain health known
research	lookup_docs, memory search (CLI)	APIs verified before writing code
develop	score_file, quick_check	Every edit gated as it lands
validate	validate_changed, security_scan	Batch gate and judges pass

Stage	Primary tools	Done when
verify	checklist, memory save (CLI)	No required step skipped

QUALITY-GATE SERVER — QUALITY PIPELINE INTERACTION

Level 2 · Deterministic checkers run in-process; no LLM in the scoring path.



Solid = MCP tool dispatch · judges run subprocess in consumer cwd.

Figure — Scoring and judges run on separate paths. Add shell and grep judges in document consumers without touching ScoreEngine.

Preset tool bundles

Use the six-server NLT fleet — nlt-build, nlt-memory, nlt-setup, nlt-linear-issues, nlt-project-docs, nlt-release-ship — as role-scoped presets that trim the tool surface under Cursor's 40-tool limit. Each runs as one shared long-lived HTTP server on a fixed port (8760-8765, ADR-0024), not a per-window stdio spawn. The canonical tool list lives in server.py::ALL_TOOL_NAMES (quality-gate server 3.12.90) — check it after every upgrade before trusting a preset.

Part IV — Tool families

PART IV

MCP tool surface

Scan the overview first: 44 tools in 9 families. Each card lists every tool with its purpose and the exact moment in the session to call it.

Family	Tools	Theme
Session bootstrap	session_start, server_info, doctor, usage, session_end, handoff_save, session_notes	First calls in every IDE session — server version, checker env, brain_bridge_health.
Scoring and gates	quick_check, score_file, quality_gate, validate_changed	Deterministic Python quality — ruff, mypy, bandit, radon, vulture, pylint, pip-audit.
Security and dependencies	security_scan, dependency_scan, dead_code, dependency_graph	Dedicated scans that go deeper than quick_check's basic pass.
Docs and impact	lookup_docs, impact_analysis, validate_config, call_graph, diff_impact, research, domain_playbook	Prevent hallucinated APIs and blind refactors.
Pipeline orchestration	checklist, init, upgrade, set_engagement_-level, pipeline, decompose, report	Checklist, init, upgrade, engagement — consumer repo scaffolding.
Linear cache-first reads	linear_snapshot_get, linear_snapshot_put, linear_list_issues, release_update, linear_-snapshot_invalidate, linear_count	Route writes through the linear-issue skill; run list reads cache-first via snapshots.
Audit campaigns	audit_campaign, audit_close_coverage, finding_to_story	Repo-wide quality campaigns and finding closure.
Metrics and telemetry	dashboard, stats, feedback	Usage dashboards and feedback signals for tuning the pipeline itself.
Memory bridge (nlt-memory profile)	memory, brain_propose_hive_elevation, brain_approve_hive_elevation	Slim memory facade onto NLT Memory and the hive-elevation safety gate.

Session bootstrap

First calls in every IDE session — server version, checker env, brain_bridge_health.

Tool	Purpose	When
session_start	Project context + pipeline hint	FIRST call each chat
server_info	Version and installed checkers	Diagnostics without full bootstrap
doctor	MCP + brain + tool drift	After upgrade or auth failures
usage	Per-session tool gaps	When checklist reports missed steps
session_end	Close feedback loop on session's brain events	LAST call each chat
handoff_save	Atomic session-handoff.md write + brain mirror	Before context runs out
session_notes	Short-lived per-session KV scratch store	Mid-session working notes

Scoring and gates

Deterministic Python quality — ruff, mypy, bandit, radon, vulture, pylint, pip-audit.

Tool	Purpose	When
quick_check	Score + gate + basic security	After each Python edit
score_file	Seven-category score	Deep review or CI report
quality_gate	Pass/fail vs preset threshold	Before declaring file done
validate_changed	Batch gate + optional judges	End of multi-file work

Security and dependencies

Dedicated scans that go deeper than quick_check's basic pass.

Tool	Purpose	When
security_scan	Bandit + secrets	Security task_type or sensitive edits
dependency_scan	pip-audit CVEs	Before release
dead_code	vulture unused symbols	Refactors and epic boundaries
dependency_graph	Import coupling	Large module changes

Docs and impact

Prevent hallucinated APIs and blind refactors.

Tool	Purpose	When
lookup_docs	Context7-backed library docs	Before any external API use
impact_analysis	Direct + transitive dependents	Before layout/engine API edits
validate_config	Docker, CI, YAML manifests	Infra or brand/template YAML
call_graph	Function callers, callees, call chains	Before function-level refactors
diff_impact	Ranked affected tests for changed files	After edits, before validate_changed
research	Unified research front door (code/web/docs)	Domain questions beyond library docs
domain_playbook	Bundled domain checklist + tool order	Security/testing/frontend domain work

Pipeline orchestration

Checklist, init, upgrade, engagement — consumer repo scaffolding.

Tool	Purpose	When
checklist	Required tools by task_type	FINAL verification step
init	Bootstrap AGENTS.md, hooks, skills	New consumer repo
upgrade	Refresh scaffolding from latest	After a quality-gate server release
set_engagement_level	high / medium / low enforcement	Operator tuning
pipeline	Runs session_start through gate in one call	Single-shot scripted runs
decompose	Breaks a vague task into ordered subtasks	Planning ambiguous work
report	Quality report, single-file or project-wide	CI report or deep review

Linear cache-first reads

Route writes through the linear-issue skill; run list reads cache-first via snapshots.

Tool	Purpose	When
linear_snapshot_get	Cache-first issue slice	Before list_issues
linear_snapshot_put	Populate cache after fetch	Immediately after list
linear_list_issues	Gate-checked list wrapper	When cache miss
release_update	Release note body from CHANGELOG	linear-release-update skill
linear_snapshot_invalidate	Evicts cached Linear snapshots	After a write changes issue state
linear_count	Open/done counts from cached snapshots	Backlog size checks

Audit campaigns

Repo-wide quality campaigns and finding closure.

Tool	Purpose	When
audit_campaign	Batch audit orchestration	Quality sweeps
audit_close_coverage	Link fix SHA to finding	After audit fix lands
finding_to_story	Finding → Linear story	Audit remediation loop

Metrics and telemetry

Usage dashboards and feedback signals for tuning the pipeline itself.

Tool	Purpose	When
dashboard	Multi-section usage/gate metrics dashboard	Weekly platform health review
stats	Per-tool call counts and success rates	Auditing tool usage patterns
feedback	Records thumbs-up/down on last tool response	After a notably good or bad verdict

Memory bridge (nlt-memory profile)

Slim memory facade onto NLT Memory and the hive-elevation safety gate.

Tool	Purpose	When
memory	Cross-session pattern save/recall facade	nlt-memory profile only
brain_propose_hive_elevation	Propose a memory key for hive elevation	Pattern worth sharing across projects
brain_approve_hive_elevation	Approve a pending hive elevation proposal	Operator review of a proposal

Part V — Quality presets and engagement

PART V

Scoring presets and enforcement

Preset	Behavior
standard	Default gate — test coverage weighted on all Python
report_authoring	Same 70/0/0 gate thresholds as standard — defaults narrative_path_globs to reports/**, zeroing test-coverage weight there
strict	Higher thresholds for library/engine code
framework	Framework-code profile — stricter API-surface weighting

Engagement levels

Switch enforcement with set_engagement_level (high / medium / low). The level changes which checklist tools are required per task_type and which lifecycle hooks get wired into the consumer IDE.

Level	document task	feature task	Hooks
high	validate_changed + checklist	score_file + quality_gate + security_scan	10 lifecycle hooks; linear cache gate block mode optional
medium	validate_changed	score_file + quality_gate	8 hooks; linear cache gate warn; pre-commit optional
low	validate_changed	quality_gate	SessionStart only; judges advisory

Checklist task types

task_type	Required	Recommended
feature	score_file, quality_gate	security_scan
bugfix	score_file	quality_gate, security_scan
refactor	score_file, quality_gate	impact_analysis, call_graph, dead_code
security	security_scan, quality_gate	score_file, dependency_scan
document	validate_changed	validate_config, lookup_docs, checklist
review	score_file, security_scan, quality_gate	checklist, dead_code
epic	checklist	score_file, quality_gate
release	release_update	dependency_scan

The **document** task_type requires validate_changed — always pass explicit file_paths, and configure blocking judges in PDF/HTML consumers. Part VI shows the ReportLab reference config.

Part VI — Document output judges

PART VI

Gating document regressions

Python quality gates do not catch thin PDF pages, missing link annotations, or a dropped --audit flag in the site prebuild. Gate those regressions with validate_changed.judges in the project quality config — pytest, grep, shell, or exists types.

validate_changed.judges — ReportLab reference

Gate **output quality** in document-shipping consumers, not only Python scores. Copy the ReportLab pattern: declare shell and grep judges under validate_changed.judges in the project quality config:

Judge	Type	Blocks when
scripts/run-pdf-judge-tests.sh	shell	uv pytest PDF regression fails
build_briefs_pdfs.py contains --audit	grep	Prebuild drops post-build audit

Scope each judge with **when_changed** globs so README-only diffs skip pytest. Set quality_preset: report_authoring to stop bogus coverage failures on reports/** narrative modules, and memory.profile: document-builder so sessions recall audit profiles and env roots automatically.

Part VII — Hooks and agent skills

PART VII

Platform automation

Platform hooks and agent skills

init writes Cursor / Claude Code hooks at medium or high engagement — no manual wiring. Skills bundle repeatable pipelines that agents invoke by name; prefer them over raw tool sequences.

Artifact	Trigger	Effect
Before-MCP hook	beforeMCPExecution (Cursor)	Log MCP tool invocations
After-edit hook	afterFileEdit (Cursor) / PostToolUse (Claude)	Remind quick_check
Stop hook	Session end	Telemetry if validate_changed skipped
Finish-task skill	Agent skill	validate_changed + checklist + memory
Handoff-session skill	Agent skill	session-handoff.md + session_end
linear-read	Agent skill	snapshot_get before list_issues
linear-issue	Agent skill	docs_validate before save_issue

Shared HTTP MCP fleet (ADR-0024)

Six long-lived HTTP MCP servers run on fixed localhost ports, started once per machine with the server CLI's fleet start — not one stdio child per Cursor window per server. Consumer repos opt in with mcp_transport: http and identify themselves per request via a project-root request header, so one fleet serves every repo and window.

Server	Port	Endpoint
nlt-build	8760	http://127.0.0.1:8760/mcp

Server	Port	Endpoint
nlt-memory	8761	http://127.0.0.1:8761/mcp
nlt-setup	8762	http://127.0.0.1:8762/mcp
nlt-linear-issues	8763	http://127.0.0.1:8763/mcp
nlt-project-docs	8764	http://127.0.0.1:8764/mcp
nlt-release-ship	8765	http://127.0.0.1:8765/mcp

Supervision is load-bearing. fleet start launches the six servers as plain subprocesses that stay in the invoking unit's cgroup, so the canonical fleet service unit is Type=oneshot with RemainAfterExit=yes — the unit stays active (exited) instead of tearing its cgroup down and killing the servers. The watchdog (a fleet-watch service + timer, every 60s) never calls fleet start directly — a plain oneshot watchdog would spawn the servers into its own cgroup and reap them on exit. It runs fleet ensure, which probes health and only restarts the canonical unit via systemctl --user restart when unhealthy.

> "The fleet install-systemd command is the single source of truth for all three units — service, watchdog service, watchdog timer. Do not hand-author them."
>
> **— ADR-0024**

Part VIII — CI and doctor

PART VIII

Shipping discipline

CI, doctor, and checker environment

Scoring runs seven checkers when they are installed in the MCP host environment: ruff, mypy, bandit, radon, vulture, pylint, pip-audit. Remember the MCP server process is not the consumer project venv — when host pytest is missing, route the judge through a shell script that calls uv run pytest.

Surface	Command	When
CI quality workflow	validate-changed --quick on PR	Python changes
.githooks/pre-commit	validate-changed on staged .py	install_git_hooks: true
doctor --quick	Brain + MCP without version drift	After upgrade
doctor	Full checker version audit	Cold start / CI image build

upgrade_skip_files — fixed vocabulary, not a glob

upgrade_skip_files in the project quality config protects generated artifacts from upgrade rewrites, but each entry must match one of a fixed set of tokens exactly — it is not a path glob, and directory tokens have no per-file granularity.

Token	Protects
agents_md	AGENTS.md
claude_md	CLAUDE.md
tech_stack_md	TECH_STACK.md
claude_settings	.claude/settings.json
claude_hooks	.claude/hooks (whole directory)
claude_agents	.claude/agents (whole directory)
claude_skills	.claude/skills (whole directory)
mcp_config	.mcp.json

> "TAP-6499: a consumer configured four full file paths under .claude/skills/ — none of which are tokens — and two upgrades silently overwrote a customized skill. Use the claude_skills directory token, or fold the customization upstream so upgrade regenerates it correctly."
>
> **— Known footgun**

Part IX — Operations

PART IX

Runbook and troubleshooting

Troubleshooting matrix

Symptom	Likely cause	Fix
judges_passed=false, pytest not found	MCP host lacks consumer venv	shell judge → uv run pytest script
brain_bridge_health.ok=false	Auth or brain down	Memory auth token; doctor
Gate fail on reports/story.py only	Coverage weight on narrative	quality_preset: report_authoring
list_issues cache gate warn	Snapshot miss	linear-read skill or snapshot_put
PDF audit fails thin pages	Story sparse or wrong profile	build --audit with template profile

> "Code gate ≠ document gate. Python can score 100 while PDFs lose link annotations or drop --audit in prebuild — configure judges explicitly."
>
> **— ReportLab quality pipeline evaluation (2026-06)**

Operator runbook

Daily agent session — run the five calls in order; skip none.

1. session_start() first call of the chat → 2. lookup_docs before touching any library API → 3. quick_check after each Python edit → 4. validate_changed(file_paths=...) with explicit paths → 5. checklist(task_type=...) as the final gate.

After a quality-gate server release — refresh every consumer.

The dev monorepo deploys blue/green (ADR-0019): the server CLI's deploy-local builds an immutable release venv under releases/{version}-{sha}/ in the server's per-user state directory, waits on a flock deploy lock and a pytest quiescence gate, then flips the current symlink atomically. Already-running servers stay pinned to their old release dir until MCP reload; only new launches pick up current.

Then refresh every consumer: run the server CLI's upgrade --force --host auto → restart the MCP host → doctor --quick to confirm version and brain health → upgrade in each consumer repo → rerun judges and confirm they still pass.

Document / PDF ship (ReportLab) — audit before deploy.

Run cd apps/docs && npm run build:pdfs (builds every volume with --audit) → npm run deploy:briefs (worktree-isolated NLTWeb push) → spot-check the downloads on briefs.nltlabs.ai before announcing.

Cross-volume links

Brief	Relationship
NLT Memory Deep Dive	Runtime recall — BrainBridge HTTP (memory via CLI, not in-process embed)
Report Studio Builder Ops	PDF engine — consumers pin nlt-report-studio tag
stakeholder and engineering briefs	Stakeholder/reference PDFs built with report-studio --audit
Server AGENTS.md	Living tool reference — quality-gate server 3.12.90 + docs-mcp companion

Colophon

Generated by Report Studio in the ReportLab checkout. Tool families mirror the quality-gate server 3.12.90 surface — verify the live tool count against AGENTS.md in each consumer repo before citing this brief.

NLT Labs · New Logic Tech
NLT Quality Pipeline — Builder Ops
Issued 2026-09-25 · 44 tools · 9 families