NLTlabs
NEW LOGIC TECH

NLT Platform Integration Map

NLT Scout · NLT Engine · AgentForge · NLT Memory

Four components. Zero turf wars.
Trust boundaries before deep dives.

AUDIENCE Architects · technical leads

Executive summary

4 COMPONENTS Scout · Engine · AF · Memory	4 ROOTS PRESENT At build time	HTTP CROSS-REPO Primary contract	Vol 4 MOTION Control-flow sequences

This volume is the static integration map architects read before Vol 5-8 deep dives. It documents where each component writes, what crosses boundaries, and how workspace roots are discovered at build time.

Key findings

1. Each component owns writes inside its trust boundary.
2. AgentForge stays consumer-agnostic — no in-tree consumer patches.
3. NLT Memory is HTTP-only through NLT Quality Pipeline bridge.

Reader path
Investors: Vol 1. Operators: Vol 2 + Vol 4. Engineers: Vol 5-8 after this map.

SECTION I

Platform overview

Component roster and integration posture.

Platform summary

NLT Labs ships four platform components that share a workspace layout but maintain strict write boundaries. NLT Scout captures ideas and portfolio artifacts. NLT Engine owns PE workflows, agent definitions, and POC ship. AgentForge runs generic orchestration. NLT Memory holds cross-session recall for every agent surface.

This volume is the integration map architects read before opening component deep dives (Vol 5-8) or communication flow sequences (Vol 4). Outward names follow docs/NLT_BRAND.md.

Sibling roots are discovered via environment variables so the same story builds on any machine with a standard checkout layout.

Component	Role	Checkout	Status
NLT Scout	Idea intake and portfolio artifacts	/home/wtthornton/NLT Scout	present

Component	Role	Checkout	Status
NLT Engine	Engine workflows, schemas, and agent definitions	/home/wtthornton/code/NLT Engine	present
AgentForge	Generic agent orchestration platform	/home/wtthornton/code/AgentForge	present
NLT Memory	Shared memory, hive, and knowledge graph service	/home/wtthornton/code/NLT Memory	present

Platform ecosystem

The four outward-facing components share a workspace layout but enforce separate write boundaries. The table below summarizes who owns durable state; Vol 3 diagrams show how compose stacks and git remotes connect.

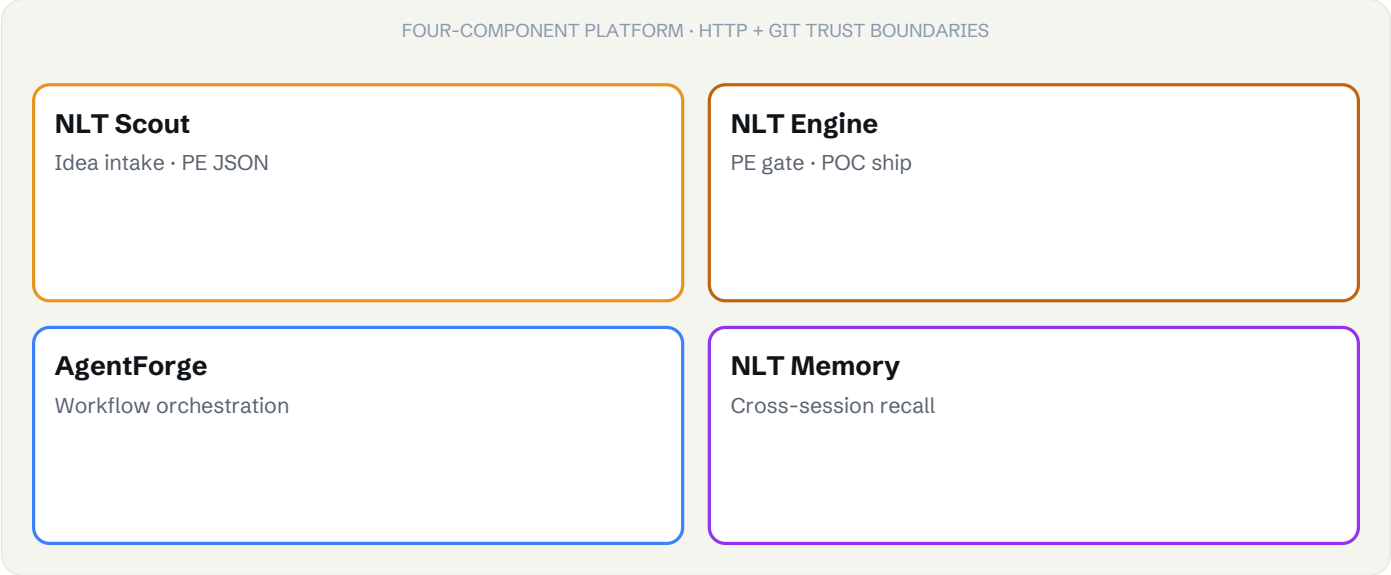


Figure 1 — Four outward-facing components and their primary roles. Deep dives: Vol 5–8.

Write vs read scope

Component	Primary writes	Primary reads
NLT Scout	ideas/, PE JSON	Intake templates, brand packs
NLT Engine	Registry, workflows, agent defs	Scout artifacts, memory recall
AgentForge	Task audit, experience index	Engine invoke, memory bridge
NLT Memory	Tiers, graph, embeddings	All MCP memory tool calls

OPERATING PIPELINE

Integration principle
Each component owns writes inside its trust boundary. Cross-repo effects use HTTP APIs or git — never in-tree patches of consumer code inside AgentForge.

Component narratives

NLT Scout

Status: present on disk. Root: /home/wtthornton/NLT Scout

NLT Scout is the pipeline entry point. Intake lands as markdown and PE JSON; dispatcher agents promote artifacts without writing into NLT Engine or AgentForge source trees.

Operators use NLT Scout when cloudflare worker intake, agentforge /projects api. Write scope stays inside ideas//*.md, pe/*.json via dispatcher agents.

- Write scope: ideas//*.md, pe/*.json via dispatcher agents
- Integrations: Cloudflare Worker intake, AgentForge /projects API

NLT Engine

Status: present on disk. Root: /home/wtthornton/code/NLT Engine

NLT Engine is the portfolio operating system. Agent folders and YAML workflows define automation; maintainers merge engine changes via PR.

Operators use NLT Engine when published schemas package, poc-builder consumer. Write scope stays inside Engine code and workflows — maintainer PRs only.

- Write scope: Engine code and workflows — maintainer PRs only
- Integrations: Published schemas package, poc-builder consumer

AgentForge

Status: present on disk. Root: /home/wtthornton/code/AgentForge

AgentForge stays consumer-agnostic. Per-project directories hold NLT Scout and NLT Engine state while the platform binary stays pure.

Operators use AgentForge when fastapi /tasks, per-project state, mcp bridges. Write scope stays inside Platform code only — no consumer-specific patches.

- Write scope: Platform code only — no consumer-specific patches
- Integrations: FastAPI /tasks, per-project state, MCP bridges

NLT Memory

Status: present on disk. Root: /home/wtthornton/code/NLT Memory

NLT Memory is the cross-session authority. The NLT Quality Pipeline memory bridge reaches it over HTTP; AgentForge sessions index outcomes for later recall.

Operators use NLT Memory when NLT Quality Pipeline memory bridge, agentforge session hooks. Write scope stays inside Brain schemas and Postgres — HTTP consumers read/write via API.

- Write scope: Brain schemas and Postgres — HTTP consumers read/write via API
- Integrations: NLT Quality Pipeline memory bridge, AgentForge session hooks

Workspace layering model

The same triage model AgentForge uses internally maps onto the NLT workspace: external systems stay outside; consumers hold credentials and domain logic; AgentForge owns orchestration, memory bridge hooks, and security boundaries.

- External systems — Linear, GitHub, Firecrawl, Exa, Tavily, YouTube, HN. AgentForge holds no opinion; consumers encode integrations.
- Consumers — NLT Engine and NLT Scout express pipelines as workflow YAML and AGENT.md configs. Consumer business logic never lands in platform trees.
- AgentForge — task pipeline, workflow DAG executor, audit log, MCP bridges. Generic across every consumer project registered via POST /projects.

Layering is enforced in CI: consumer patches inside AgentForge platform trees fail review; memory bridge calls stay on HTTP.

Layering invariant
Consumers call the platform; the platform never imports a consumer. Git and HTTP are the only durable cross-boundary handoffs.

Workspace portfolio roster

Project	Outward name	Role	Relationship to AgentForge
AgentForge	AgentForge	Agent runtime / OS	Substrate — lifecycle, pipeline, memory bridge, platform purity CI.
NLT Engine	NLT Engine	PE → POC → portfolio	Consumer + publication channel; drives AgentForge over HTTP.
NLT Scout	NLT Scout	Idea mining	Strict consumer registered via POST /projects; workflows on platform.
NLT Quality Pipeline	NLT Memory	Memory bridge	HTTP MCP tools into NLT Memory — never in-process embed in consumers.

Live workspace snapshot

Build-time adapters found 36 agents and 19 workflows across sibling checkouts. Missing roots yield empty tables — not silent placeholder copy.

Agent roster (live)

Agent	Repo	Model	Brain	Summary
banker	NLT Engine	opus	yes	Banker — prepares the portfolio LLC's business banking application via Mercury. HIGH-risk: the agent prepares, a human s
compliance-baseline	NLT Engine	opus	yes	Compliance Baseline — drafts the privacy policy, terms of service, and cookie-banner copy for a newly-formed portfolio.
domain-and-brand	NLT Engine	sonnet	yes	Domain & Brand — registers the portfolio's .com via Cloudflare, points DNS at a placeholder, and generates the brand pac
gtm-launcher	NLT Engine	sonnet	yes	GTM Launcher — productizes the publish-to-prod path. Takes a CreatedPortfolio, publishes a live landing page on the regi
gtm-operator	NLT Engine	sonnet	yes	GTM Operator — converts the POC director's brief into a runnable go-to-market motion. Emits GTM-EXECUTION.md with the 18
incorporator	NLT Engine	opus	yes	Incorporator — forms the portfolio LLC via Stripe Atlas. HIGH-risk: every entity-type and jurisdiction choice waits for
linear-intake-create	NLT Engine	sonnet	no	Linear Intake Creator — given a new intake.md from nlt-portfolio (form- or relay-originated), creates a corresponding Li
market-pricing-researcher	NLT Engine	sonnet	yes	Market Pricing Researcher — independent comparable-set + BOM-multiple + modal-price pricing recommendation for hardware
number-auditor	NLT Engine	sonnet	yes	Pipeline number-hygiene auditor — verifies every market-sizing claim, unit-economics line, and revenue projection has a
pe-competitive-analyst	NLT Engine	sonnet	yes	PE pipeline competitive analyst — maps competitors and assesses moat for a proposed idea.
pe-creative-director	NLT Engine	sonnet	yes	PE pipeline creative director — names the venture, defines positioning and brand.
pe-devils-advocate	NLT Engine	sonnet	yes	PE pipeline devil's advocate — surfaces fatal flaws and worst-case scenarios.
pe-feasibility-analyst	NLT Engine	sonnet	yes	PE pipeline feasibility analyst — evaluates execution realism, technical risks, and timeline.
pe-financial-modeler	NLT Engine	sonnet	yes	PE pipeline financial modeler — projects revenue, costs, and breakeven for a proposed idea.
pe-firm	NLT Engine	sonnet	yes	PE pipeline decision-maker — aggregates researcher briefs into fund/pass/dig verdict.
pe-market-researcher	NLT Engine	sonnet	yes	PE pipeline market researcher — sizes TAM, SAM, SOM and growth drivers for a proposed idea.

Agent	Repo	Model	Brain	Summary
pe-regulatory-analyst	NLT Engine	sonnet	yes	PE pipeline regulatory analyst — surfaces compliance risks and requirements.
poc-builder	NLT Engine	fable	yes	POC Frontend Builder — implements the Astro/Tailwind site from design spec and copy, generates renders (hardware), commi
poc-builder-gateway	NLT Engine	sonnet	no	Thin AF gateway for the poc-ship build step — HMAC-signs and dispatches the build to NLT's HTTP build service, polls to
poc-content-qa	NLT Engine	sonnet	yes	POC Content QA Agent — independent sales-readiness reviewer for every customer-facing + investor-facing page poc-copywri
poc-copywriter	NLT Engine	sonnet	yes	POC Messaging Strategist — writes every word on the POC site: headlines, feature copy, CTAs, pricing copy, /inside inves
poc-designer	NLT Engine	sonnet	yes	POC Brand Designer — turns the Venture Designer's brief into a pixel-precise design specification: layout, typography, c
poc-director	NLT Engine	opus	yes	POC Venture Designer — classifies funded idea, runs competitive research, names the project, writes the master POC brief
poc-gtm-qa	NLT Engine	sonnet	no	POC GTM QA — deterministic scorer for gtm_operator section population. Drives the gtm_regen loop until ≥16/18 sections a
poc-post-deploy-audit	NLT Engine	sonnet	yes	POC Post-Deploy Audit — operator-grade live-URL audit run automatically after every poc-ship build. Shares check logic w
poc-product-designer	NLT Engine	sonnet	yes	POC Product Designer — industrial design thinking for hardware POCs only. Derives form from mechanism, mechanism from us
poc-quality-reviewer	NLT Engine	sonnet	yes	POC Quality Reviewer — unified ship-readiness verdict. Aggregates visual_qa, content_qa, market_pricing, and builder lin
poc-visual-assets	NLT Engine	opus	yes	POC Visual Assets Agent — autonomously generates 4 photorealistic product renders (studio, hero, product-family, explode
poc-visual-qa	NLT Engine	opus	yes	POC Visual QA Agent — independent cross-family vision judge for every render `poc-visual-assets` produces. Uses Claude S
candidate-surfacer	NLT Scout	sonnet	no	Files high-confidence promoted IdeaCandidates as Linear issues for human review. Deduplicates by candidate URL. Requires
categorize	NLT Scout	haiku	no	Taxonomy guidance for IdeaCandidate categorization in NLT Scout. The deterministic tagger scores title+summary aga

Agent	Repo	Model	Brain	Summary
general-web-search	NLT Scout	sonnet	no	General-web discovery agent. Searches recent open-web software-idea signals (launches, Show-HN-style posts, indie-hacker
intake-submitter	NLT Scout	sonnet	no	Submits promoted IdeaCandidates as PE-intake issues in the NLT Idea Intake Linear project, in the portfolio submit form'
market-signal	NLT Scout	sonnet	no	Evidence-backed market-signal pre-filter for NLT Scout. Firecrawl-fetches the source URL, Tavily Research synthesi
skeleton-probe	NLT Scout	sonnet	no	Minimal probe agent that proves the NLT Scout AgentForge publish loop end-to-end.
thesis-writer	NLT Scout	sonnet	no	Writes a 2-3 paragraph investor thesis for a promoted NLT Scout idea, grounded in the market-signal assessment (su

Workflow manifests chain the roster above into source-vertical DAGs — search or scrape nodes, extraction agents, categorize, export. The inventory below lists every published workflow slug with node counts and primary agent bindings discovered at build time.

Workflow inventory (live)

Workflow	Repo	Nodes	Agents
create-portfolio	NLT Engine	3	nlt-incorporator, nlt-banker, nlt-domain-and-brand
linear-intake-create	NLT Engine	1	nlt-linear-intake-create
pe-evaluate-test	NLT Engine	9	nlt-pe-market-researcher, nlt-pe-competitive-analyst, nlt-pe-feasibility-analyst, nlt-pe-financial-modeler, nlt-pe-regulatory-analyst, nlt-pe-creative-director, nlt-pe-devils-advocate, nlt-number-auditor, nlt-pe-firm
pe-evaluate	NLT Engine	9	nlt-pe-market-researcher, nlt-pe-competitive-analyst, nlt-pe-feasibility-analyst, nlt-pe-financial-modeler, nlt-pe-regulatory-analyst, nlt-pe-creative-director, nlt-pe-devils-advocate, nlt-number-auditor, nlt-pe-firm
poc-ship-test	NLT Engine	12	nlt-poc-director, nlt-market-pricing-researcher, nlt-poc-product-designer, nlt-poc-visual-assets, nlt-poc-designer, nlt-poc-copywriter, nlt-gtm-operator, nlt-poc-visual-qa, nlt-poc-content-qa, nlt-poc-builder-gateway, nlt-poc-post-deploy-audit, nlt-poc-quality-reviewer
poc-ship	NLT Engine	13	nlt-poc-director, nlt-market-pricing-researcher, nlt-poc-product-designer, nlt-poc-visual-assets, nlt-poc-designer, nlt-poc-copywriter, nlt-gtm-operator, nlt-poc-gtm-qa, nlt-poc-visual-qa, nlt-poc-content-qa, nlt-poc-builder-gateway, nlt-poc-post-deploy-audit, nlt-poc-quality-reviewer
test-echo	NLT Engine	1	general
flippa	NLT Scout	1	general-web-search
general-web	NLT Scout	1	general-web-search
hn-vertical	NLT Scout	1	—
indie-hackers	NLT Scout	1	general-web-search
product-hunt	NLT Scout	1	general-web-search
reddit	NLT Scout	1	general-web-search
reviews	NLT Scout	1	general-web-search
skeleton	NLT Scout	1	skeleton-probe
socrata-procurement	NLT Scout	1	—
submit	NLT Scout	1	intake-submitter
surface	NLT Scout	1	candidate-surfacer
youtube	NLT Scout	1	general-web-search

Production workflows — pipeline order

Workflow	Nodes	on_error	Pipeline (node order)
create-portfolio	3	fail	incorporator → banker → domain_and_brand
linear-intake-create	1	partial	create
pe-evaluate	9	partial	market → comp → feas → fin → reg → creative → dev → auditor → firm
poc-ship	13	partial	director → market_pricing → product_designer → visual_assets → designer → copywriter → gtm_operator → gtm_qa → visual_qa → content_qa → builder → post_deploy_audit → quality_reviewer
test-echo	1	partial	echo

NLT Engine — consumer boundary

NLT Engine (checkout NLT Engine) is both an AgentForge **consumer** and the **publication channel** for portfolio.nltlabs.ai. It drives PE evaluate, poc-ship, and portfolio-create workflows over HTTP without editing AgentForge compose, bind-mounts, or platform environment.

Engine write scope

Concern	What NLT Engine owns
Domain	PE rubric, POC build pipeline, portfolio generation, brand/copy linting.
Integration	Linear intake dispatch, GitHub publication, content-safety gating.
Platform calls	Repo registration, webhook secrets, workflow dry-runs, brief extraction.
Persistence	Atomic JSON (decision.json, briefs) + research library migrations.

intake_dispatcher polls Scout artifacts and queues pe-evaluate work. Publish scripts push topology and portfolio artifacts. Crash-safe atomic writes keep decision.json and brief files consistent for Linear sweeps and portfolio renderers.

Consumer + channel
Engine drives AgentForge to execute; it renders FUND proof URLs without forking platform Python.

NLT Scout — consumer boundary (summary)

NLT Scout is an AgentForge project (slug: NLT Scout), not a standalone service. The repo supplies domain models, taxonomy, and per-source extraction logic; AgentForge supplies orchestrator, scheduler, ingest, and dedup. Full vertical depth lives in Vol 7.

Scout capability map

Capability	Status
Domain models (Source · RawItem · IdeaCandidate)	Shipped in Scout checkout
DuckDB local storage adapter	Shipped
NLT v1 taxonomy loader	Shipped
AgentForge project + workflow round-trip	Shipped
HN vertical (kind: http node)	Depends on AgentForge http primitive

Workspace boundary
When a platform primitive is missing, Scout files upstream work — it does not fork a parallel HTTP path inside the consumer repo.

SECTION II

Trust boundaries

Write scopes and cross-repo contracts.

Trust boundaries

Each component maintains its own write boundary. Reads may cross repos; durable writes stay inside the owning system unless mediated by git or a published HTTP API.

When onboarding a new checkout, verify env roots first (Section V), then confirm the matrix in Section IV matches your machine layout.

TRUST BOUNDARIES — who may write what

nlt-portfolio
Artifacts only — ideas/<slug>/.md, pe/*.json*

- ✓ Cloudflare Worker (intake.md)
- ✓ intake_dispatcher (pe/*, decision.md)
- ✓ poc-builder (poc/, portfolio-entry.json)
- ✓ Operator (human PR)

nlt-engine
Engine code, workflows, schemas, agents

- ✓ Maintainers (PR + merge)
- ✗ Agents (never push to engine code)
- ✓ Schemas via nlt-engine-schemas package

AgentForge
Generic platform — no consumer code

- ✓ Platform maintainers only
- ✗ Consumer-specific code (rejected by platform-purity rule)
- ✓ Per-project state via /projects API

tapps-brain
Memory service — Postgres, schemas, hive

- ✓ tapps-brain maintainers
- ✗ AgentForge (HTTP-only consumer)
- ✓ Schema migrations owned by brain compose

Each repo writes only to itself. Cross-repo writes happen via published HTTP + git.

Write scope by component

Component	Owns writes	Cross-boundary contract
NLT Scout	ideas/, intake markdown	Git push to Engine consumer
NLT Engine	Registry, workflows, agents	HTTP invoke to AgentForge
AgentForge	Audit log, task state	HTTP to NLT Memory bridge
NLT Memory	Embeddings, tiers, graph	Published HTTP API only

- Never patch consumer code into AgentForge platform trees.
- NLT Memory is never embedded in-process in consumer repos.
- Git is the handoff for PE JSON and idea markdown.
- HTTP is the handoff for invoke, recall, and reinforce.

Boundary concerns by component

Component	Owns	Never does
NLT Scout	Intake markdown, idea taxonomy, source adapters	Patch AgentForge backend/ or write Engine registry directly
NLT Engine	PE rubric, workflows, portfolio publish, decision.json	Edit AgentForge compose, bind-mounts, or platform env
AgentForge	Orchestrator, audit log, MCP bridges, /projects state	Import consumer repos or encode Scout/Engine domain rules
NLT Memory	Embeddings, tiers, graph index, HTTP API	Ship inside consumer Python processes

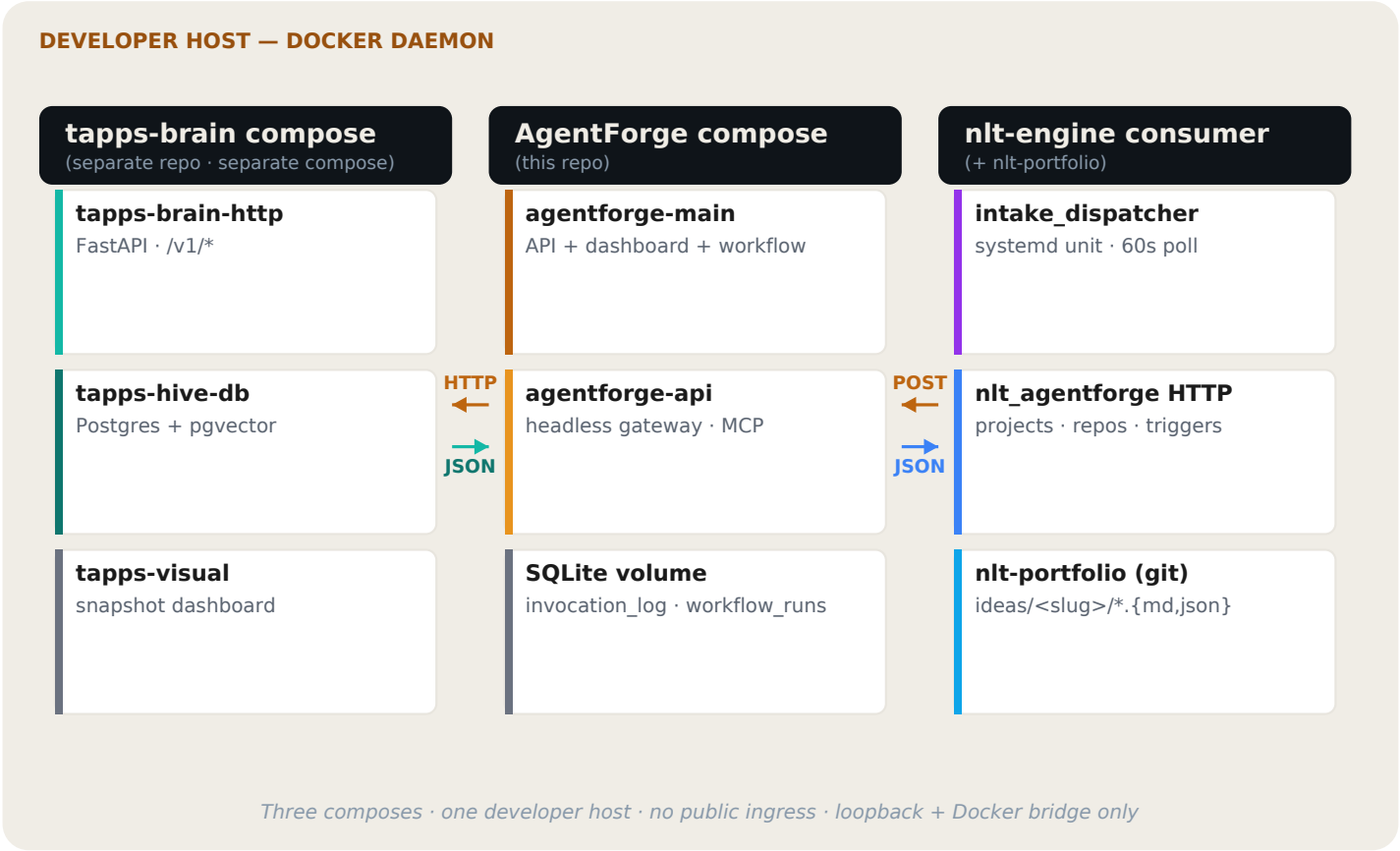
SECTION III

Component map

Spatial layout of platform siblings.

Production topology shows how AgentForge compose stacks connect to NLT Engine consumers and the NLT Memory sidecar. Compose service labels use internal names; outward names are in the table above.

Local development often runs a subset of these services. Missing roots at build time appear as “missing (set env root)” in the overview table — not as silent failures in PDF copy.



Compose labels → outward names

Compose service	Outward name	Typical port / surface
agentforge-net	AgentForge	Workflow HTTP invoke
tapps-hive-db	NLT Memory	Postgres + pgvector
nlt-builder	NLT Engine builder	OCI build substrate
brain-bridge	NLT Memory bridge	HTTP MCP bridge (NLT Quality Pipeline)

SECTION IV

Integrations

Matrix of durable cross-component links.

Integration contracts

The matrix summarizes durable integration surfaces. Cells describe what crosses each trust boundary — not every HTTP call in the platform.

From \ To	NLT Scout	NLT Engine	AgentForge	NLT Memory
NLT Scout	—	schemas	invoke	recall
NLT Engine	intake	—	agents	memory
AgentForge	project state	workflows	—	bridge
NLT Memory	—	—	HTTP API	—

Contract notes

- NLT Scout → NLT Engine: PE JSON and idea markdown via git or dispatcher.
- NLT Engine → AgentForge: workflow invoke and agent definitions over HTTP.
- AgentForge → NLT Memory: recall, experience, and reinforce via brain bridge.
- NLT Quality Pipeline → NLT Memory: session memory tools; never in-process embed.
- Missing env root: story builds with “missing” status — set NLT_*_ROOT.
- Auth failure on bridge: check TAPPS_BRAIN_AUTH_TOKEN and brain health.
- Stale recall: reinforce or supersede architectural tier entries in memory.
- Consumer patch in AF tree: rejected by platform purity review.

Failure modes

Symptom	Likely cause	Remediation
Memory search empty	Bridge auth or cold brain	tapps doctor + token
Invoke 403	Bearer / policy mismatch	Verify afp_ scope
Build missing root	Env not set	Section V env table
Gate fail on PR	Changed Python below threshold	tapps_quick_check locally

API and MCP surfaces

Integration contracts are published surfaces — not ad-hoc repo writes. Vol 4 documents motion across these surfaces; this table is the static index.

Surface	Owner	Contract	Typical caller
POST /tasks/invoke	AgentForge	Bearer + policy + audit	NLT Engine workflows
POST /projects	AgentForge	Register consumer project	Scout / Engine publish
tapps_memory *	NLT Quality Pipeline → NLT Memory	HTTP bridge + auth token	IDE agents, CI bots
MCP tool servers	AgentForge	Bridge-only tool exposure	CliRunner during nodes
Git push / PR	Each consumer	Artifact handoff	Scout → Engine intake
portfolio.nltlabs.ai	NLT Engine	Public proof URL	Investors after poc-ship

- Never embed NLT Memory in-process — TAPPS_BRAIN_AUTH_TOKEN on MCP hosts only.
- AgentForge platform purity CI rejects consumer patches under backend/.
- Consumer publish scripts push topology; they do not fork orchestrator code.

Portfolio value chain

The workspace is a funnel from raw signal to a published portfolio entry. AgentForge is the shared substrate under every stage — scale by adding AGENT.md configs and workflow YAML, not platform Python per consumer.

Stage	Producer	Consumer of	Platform service
Discover	NLT Scout	External sources	Orchestrator + ingest + dedup
Evaluate	NLT Engine	Idea candidates	Task pipeline + memory recall
Build (POC)	NLT Engine	FUND verdict output	Workflows + repo context
Publish	NLT Engine	POC artifacts	Audit + portfolio render

Portfolio thesis
One runtime, many consumers. Generic substrate; consumer-specific behaviour stays in consumer repos.

Triage playbook — adding a workspace project

Every new NLT workspace project answers the same three questions AgentForge uses internally before accepting platform code. This discipline keeps one runtime carrying an open-ended portfolio without special-case platform Python.

Three-question triage

Question	If yes →	If no →
Consumer business logic masquerading as platform?	Land in Scout / Engine / consumer agent	Evaluate generic primitive in AgentForge
Duplicates an existing platform primitive?	Extend the primitive — do not fork	Wire consumer YAML to existing node kinds
Fits reference-architecture layer?	Document in Vol 3 matrix + Vol 4 flow	Reject or redesign boundary

- Generic substrate grows in AgentForge; domain-specific mining and evaluation stay in consumer repos.
- Git artifacts and HTTP invoke are the only durable cross-boundary handoffs.
- Vol 4 documents motion once static boundaries in this volume are understood.

Production workflows on the integration map

NLT Engine publishes pe-evaluate and poc-ship to AgentForge. Vol 4 traces operator motion; this section indexes the published DAG shapes architects compare against live YAML when onboarding a new checkout.

pe-evaluate — fan-out research, aggregated verdict

Independent research analysts return typed briefs with citations. A number-auditor checks figures; pe-firm aggregates into FUND / PASS / DIG. Production manifests typically use on_error: partial so one analyst failure does not abort the entire evaluation.

pe-evaluate — published node map

Node	Kind	Agent
market	agent	nlt-pe-market-researcher
comp	agent	nlt-pe-competitive-analyst
feas	agent	nlt-pe-feasibility-analyst
fin	agent	nlt-pe-financial-modeler
reg	agent	nlt-pe-regulatory-analyst
creative	agent	nlt-pe-creative-director
dev	agent	nlt-pe-devils-advocate
auditor	agent	nlt-number-auditor
firm	agent	nlt-pe-firm

poc-ship — builder pipeline with convergence loops

After FUND, poc-ship scopes builder work across design, build, and QA nodes. Named convergence loops (e.g. visual_assets ↔ visual_qa) re-run until a judge converges or the iteration cap is hit — bounded loops at the consumer layer.

poc-ship — published node map

Node	Kind	Agent
director	agent	nlt-poc-director
market_pricing	agent	nlt-market-pricing-researcher
product_designer	agent	nlt-poc-product-designer
visual_assets	agent	nlt-poc-visual-assets
designer	agent	nlt-poc-designer
copywriter	agent	nlt-poc-copywriter
gtm_operator	agent	nlt-gtm-operator
gtm_qa	agent	nlt-poc-gtm-qa
visual_qa	agent	nlt-poc-visual-qa
content_qa	agent	nlt-poc-content-qa
builder	agent	nlt-poc-builder-gateway
post_deploy_audit	agent	nlt-poc-post-deploy-audit
quality_reviewer	agent	nlt-poc-quality-reviewer

poc-ship convergence loops

Loop	Members	Max iter	Convergence
visual_regen	visual_assets, visual_qa	3	visual_qa.converge
content_regen	copywriter, content_qa	3	content_qa.converge
gtm_regen	gtm_operator, gtm_qa	3	gtm_qa.converge

Security and auth surfaces

Integration failures in production are usually auth or scope mismatches — not missing YAML. This table indexes the credentials operators verify before tracing Vol 4 control flows.

Surface	Credential / scope	Typical failure
AgentForge invoke	Project bearer → afp_ scope	403 policy mismatch on pe-evaluate
NLT Memory bridge	TAPPS_BRAIN_AUTH_TOKEN on MCP host	Empty recall / brain_auth_failed
Linear intake	OAuth via linear MCP in Scout agents	Dispatcher skips unauthenticated writes
Portfolio publish	GitHub + Cloudflare tokens in Engine	Slug withheld after deploy audit fail
CI quality gate	NLT Quality Pipeline hooks in consumer repo	Pre-commit blocks undeclared Python edits

SECTION V

Workspace roots and env vars

Configure before building off default layout.

Sibling checkouts default to the parent of ReportLab. Override roots when building from a non-standard workspace layout.

Build portability
Set NLT_ENGINE_ROOT, AGENTFORGE_ROOT, TAPPS_MCP_ROOT, and NLT_SCOUT_ROOT when not using ~/code sibling checkouts.

Variable	Outward component	Default sibling
NLT_WORKSPACE_ROOT	—	../ (parent of ReportLab)
NLT_SCOUT_ROOT	NLT Scout	NLT Scout
NLT_ENGINE_ROOT	NLT Engine	NLT Engine
AGENTFORGE_ROOT	AgentForge	AgentForge
TAPPS_BRAIN_ROOT	NLT Memory	NLT Memory

Confidential — prepared for NLT Labs stakeholders. Outward names follow docs/NLT_BRAND.md. Internal checkout names appear only in builder ops volumes (9–10).