Vasic Digital

AI-native software engineering · Evidence-based portfolio

The Helix product family, the vasic-digital utility fleet, and the Server Factory toolchain — AI development systems built to be trusted, on a shared engineering Constitution.

140+	43	25	439
REPOSITORY FLEET	LLM PROVIDERS	LINUX DISTROS PROVISIONED	PASSING TESTS · MAIL SERVER FACTORY

1. Overview — the whole picture first

This is a single, unified portfolio used by both vasic.digital and milosvasic.ru. It describes not a scattering of side projects but a deliberately-engineered **fleet**: **large product applications** sitting on top of **dozens of small, decoupled, independently-tested modules**, the whole thing governed by a shared engineering **Constitution** and verified by an **anti-bluff QA** discipline. That structure is the differentiator. Most portfolios are a list of things that were built; this is a system in which every product is assembled from proven, reusable parts, every part is held to the same non-negotiable rules, and every advertised capability has captured evidence behind it. The dominant language is **Go**, with Kotlin/KMP, TypeScript/React, Python, Swift, and Shell used where each genuinely fits best — Go for high-throughput services and libraries, Kotlin for provisioning and cross-platform mobile, TypeScript for typed frontends, Python for AI/ML glue.

What makes the body of work cohere is that the discipline is mechanical, not aspirational. A shared Constitution ships as a Git submodule and is inherited across a 140+-repository fleet, so a single rule change propagates everywhere; an anti-bluff QA layer refuses to record a pass without runtime proof. The Helix family, the utility fleet, and the Server Factory toolchain are three expressions of one idea — build once, reuse everywhere, and prove it works before calling it done.

Everything below is presented in priority order:

1. **Governance & QA pillars** — HelixConstitution, HelixQA (the discipline that makes the rest trustworthy).
2. **The Helix product family** — the AI-development lifecycle (HelixTrack first).
3. **LLM infrastructure** — provider abstraction, orchestration, verification.
4. **vasic-digital utilities** — product-grade standalone tools.
5. **Server Factory** — infrastructure-automation heritage (ranked last).

The unifying thesis: **a feature is done only when a real user can use it and there is captured evidence to prove it.**

---

2. Governance & QA pillars

These two come first because everything else in this portfolio borrows its credibility from them. Together they turn “trust me, it works” into an auditable fact — the Constitution encodes the rules, HelixQA proves they were followed.

- **HelixConstitution** — a universal, project-agnostic engineering rulebook shipped as a Git submodule and inherited across a 140+-repository fleet. Anti-bluff evidence gates, false-positive immunity, data/host safety, coverage discipline; extend-but-never-weaken inheritance; propagation gates that grep for required clauses fleet-wide; every gate paired with a mutation test proving it isn’t a sham.

- **HelixQA** — anti-bluff QA orchestration (Go). Written YAML test banks plus fully-autonomous LLM-and-computer-vision QA sessions across Android, Android TV, Web, and Desktop; no PASS without captured evidence (screenshots, logcat, video, stack traces). The Constitution's mandated QA test-type (§11.4.169).

3. The Helix product family

The Helix line is the flagship: an interlocking family of products that spans the entire AI-development lifecycle, from planning and specification through building, memory, translation, and delivery. Each is a real product in its own right, but the design intent is that they compose — the same governance, the same reusable modules, the same evidence discipline underneath all of them.

- **HelixTrack** — a JIRA alternative for the free world; flagship of the Helix-Track line.
- **HelixAgent** — ensemble LLM service: multiple models debate and ship the answer they agree on, with verification-based provider selection.
- **HelixCode** — enterprise-grade distributed AI development platform; divides work across SSH-managed workers with automatic checkpoint/rollback; REST, CLI, TUI, MCP interfaces.
- **HelixCluster** — a distributed operating system for AI compute, from datacenter GPUs to edge handhelds, under one control plane.
- **HelixBuilder** — an AI-powered pipeline for building applications, one category at a time.
- **HelixSkills** — a governed, constitution-backed skills system for CLI AI agents (skills, MCP tool servers, Claude Code plugins).
- **HelixSpecifier** — spec-driven development that scales its ceremony to the work.
- **HelixMemory** — one memory brain for AI agents, fusing four best-in-class engines.
- **HelixTranslate** — verified-model book translation; honest by design, never a silent fallback.
- **HelixTerminator** — a zero-trust terminal platform: every SSH session secured, shared, and AI-assisted.
- **HelixGitpx** — federated Git across a dozen hosts; one source of truth, mirrored everywhere.
- **HelixOTA** — universal, decoupled over-the-air updates; zero-brick by design.
- **HelixPlay** — turn any GPU machine into your own cloud-gaming appliance.
- **Helix-Flow** — Helix-platform inference product. *UNVERIFIED / blocked on source: its public repo currently has only a one-line README; presented at product depth only when real documentation exists.*

4. LLM infrastructure

Beneath the products sits the substrate that makes them provider-agnostic and dependable: one interface over dozens of LLM providers, one control plane for headless coding agents, and one verification source of truth. This is the layer that lets everything above swap models, survive provider outages, and refuse to trust an LLM that can't prove it understands the task.

- **HelixLLM** — one binary, six modes: OpenAI- and Anthropic-compatible inference over HTTP/3, local llama.cpp, scored fallback chain, RAG pipeline, ReAct agents.

- **LLMProvider** — one interface, 43 providers, with circuit breakers, retries, and health built in.
- **LLMOrchestrator** — one control plane for every headless CLI coding agent (OpenCode, Claude Code, Gemini, Junie, Qwen Code).
- **LLMsVerifier** — verify, monitor, optimize: the single source of truth for LLM/provider/verification metadata, with a mandatory model comprehension gate.

5. vasic-digital utilities

Standalone, product-grade tools that each solve a hard problem in their own right — and, not coincidentally, exercise the reusable-module fleet in anger. They range from a resilient multi-protocol media system to a markdown-to-video course pipeline to a content-hashed doc/DB sync engine; several are frank about their maturity, and those flags are kept, not buried.

- **Catalogizer** — multi-protocol (SMB/FTP/NFS/WebDAV/local), encrypted, self-hostable media collection management; Go/Gin API + React UI; resilient offline-tolerant monitoring; built on 21 `digital.vasic.*` submodules.
- **Courses-Creator** — markdown-to-video course pipeline; multi-LLM enrichment, TTS (Bark/SpeechT5), desktop/mobile/web players; graceful no-API-key mode.
- **VisionEngine** — decoupled Go toolkit fusing classic CV with multi-provider LLM vision; navigation graphs with BFS + DOT/JSON/Mermaid export; OpenCV build-tag-gated.
- **DocProcessor** — documentation-to-feature-map with verification-coverage tracking; LLM or heuristic/offline extraction; Apache-2.0.
- **Docs Chain** — content-hashed, bidirectional, atomic doc/DB sync (Salsa-style incremental recompute over a DAG). *Phases 1–5 GREEN; Phases 6–7 PLANNED.*
- **Herald** — reliable multi-channel notifications with natural-language, three-tier intent resolution (command → LLM → clarify); first Docs Chain consumer.
- **task_bridge** — decoupled, bidirectional task/board sync (SQLite SSoT ? docs ? ClickUp). *P1 scaffold — sync logic not yet implemented.*
- **Vasic Digital Reusable Module Suite** — the `digital.vasic.*` “standard library”: infrastructure primitives, AI building blocks, and defensive-LLM guardrails, plus a Kotlin Multiplatform mirror. *Several org repos are SCAFFOLD/WIP — flagged, not shipped.*

6. Server Factory (infrastructure-automation heritage — ranked last)

Ranked last by design, not by quality: the Server Factory toolchain predates the AI line and shows where the “build once, reuse everywhere” philosophy first took shape. Its flagship — a mail server you describe in JSON and provision anywhere — is a mature, well-tested product; the supporting cast is presented at its true, varied maturity rather than dressed up.

- **Mail Server Factory** — declarative JSON → fully-provisioned, Dockerized mail server across 12 connection types and 25 Linux distributions; enterprise security; reports 439 passing tests and a clean SonarQube gate. Flagship of the Server-Factory org.
- **Server Factory Core Framework** — the shared Kotlin engine every factory builds on.

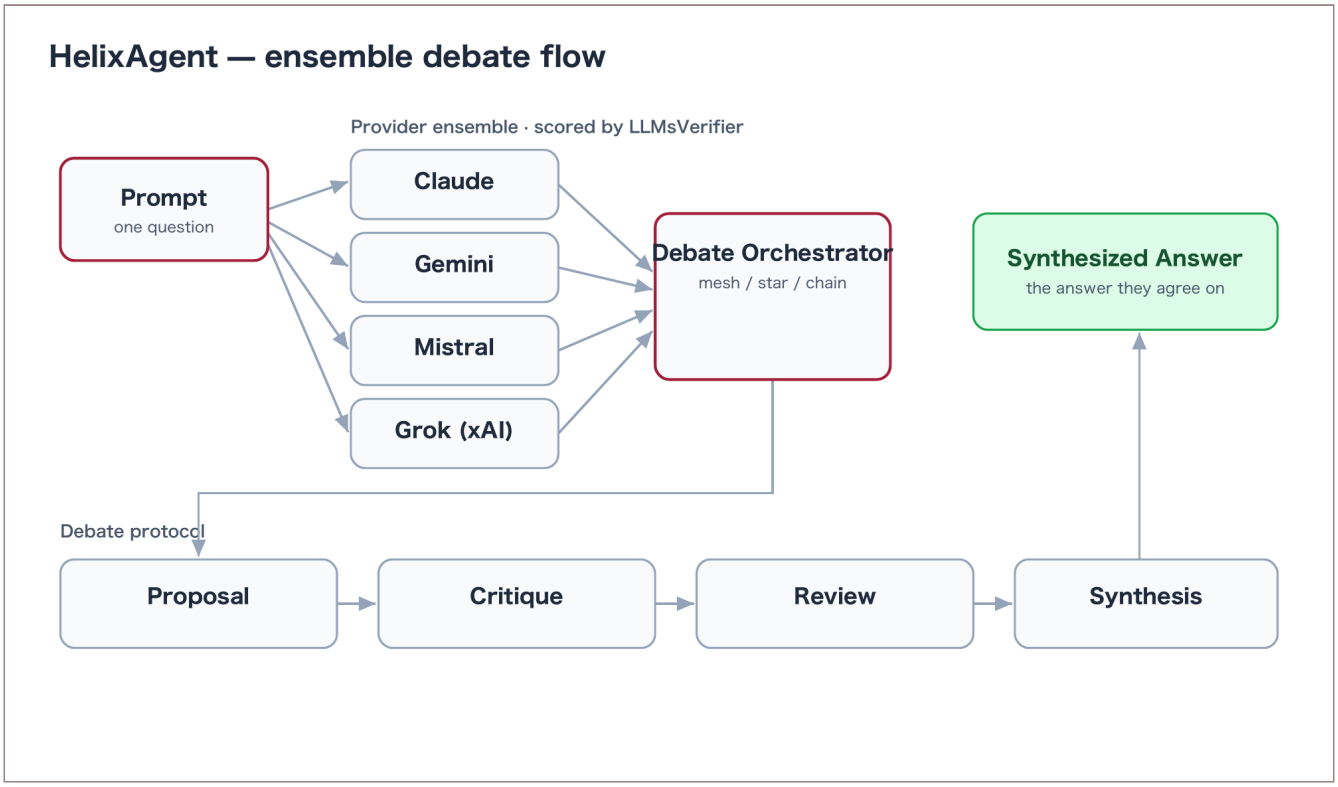
- **Qemu-Utils** — QEMU VM images managed like artifacts: download/cache/run, compress/publish, bridge/TAP networking, ISO installs; Linux + macOS.
- **Parallels-Utils** — Parallels (macOS) VM image compression, publishing, and retrieval via simple settings files.
- **Server Factory — Additional Components** — service factories (Web/SonarQube/Caching-Proxy), Definitions packs, and Utils. *Service factories are placeholder-documented — UNVERIFIED / early-stage.*

---

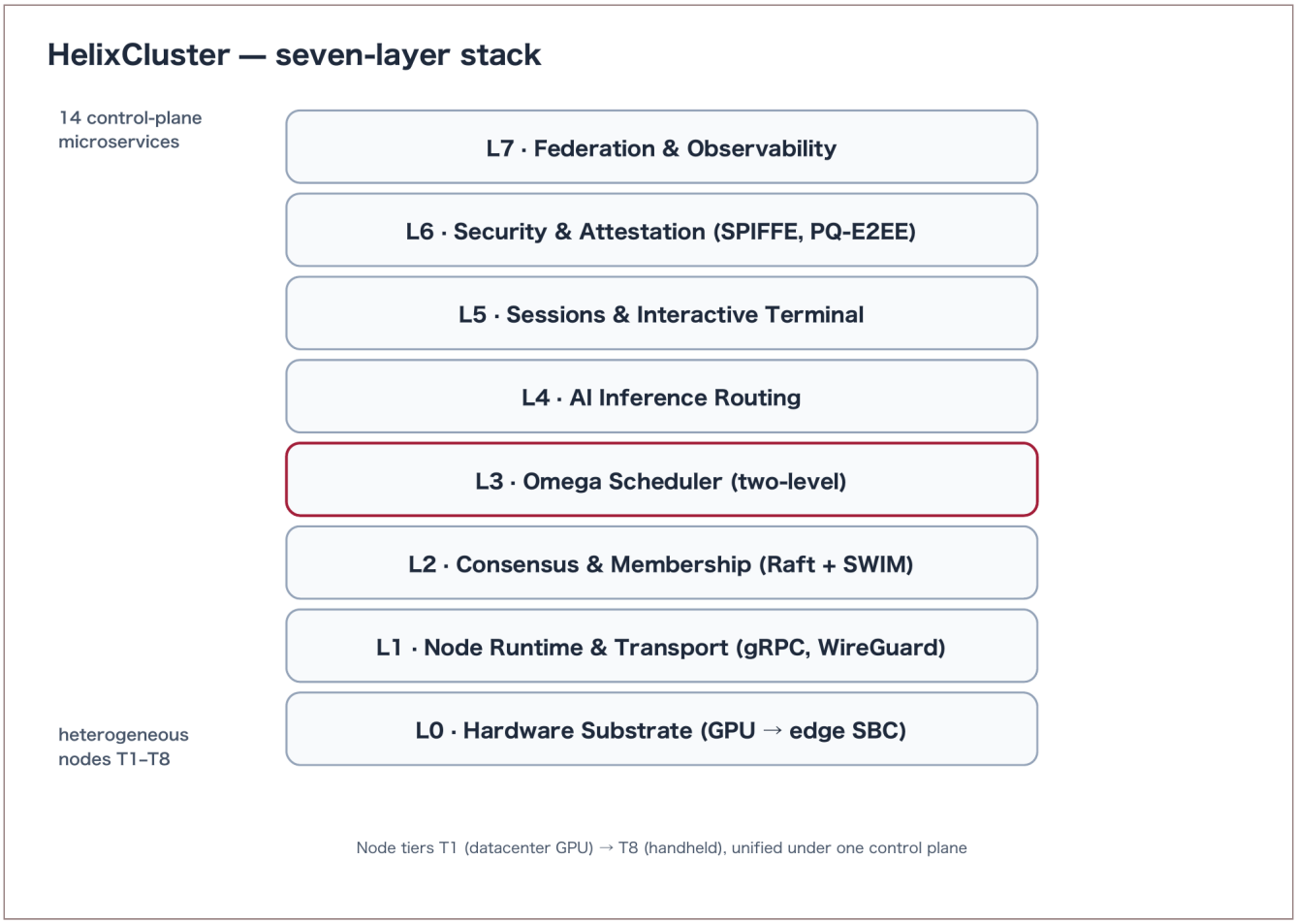
7. Technology index (evidence-based)

- **Languages:** Go (dominant), Kotlin & Kotlin Multiplatform, TypeScript, Python, Swift, Shell; PL/pgSQL; TLA+ (formal specs in helix_cluster).
- **AI / LLM:** 40+ provider access, MCP, RAG, vector DBs & embeddings, planning (HiPlan/MCTS/Tree-of-Thoughts), LLMOps, benchmarking (SWE-bench/HumanEval/MMLU), TTS (Bark/SpeechT5), computer-vision + LLM-vision, guardrails/red-team.
- **Backend:** Gin, gRPC + Protobuf, HTTP/3 (QUIC), WebSockets, Angular, React, Kafka/RabbitMQ.
- **Data:** PostgreSQL, SQLite, SQLCipher, Redis, Neo4j, ClickHouse, MinIO/S3/GCS/Azure.
- **Infra / DevOps:** Docker & Compose, Kubernetes + Helm, Prometheus + Grafana, OpenTelemetry, QEMU/Libvirt/Parallels; GitHub Actions, Gradle, Make.
- **Testing / QA:** HelixQA, Challenge harnesses with mutation gates, `go test -race`, visual-regression tooling, ADB device testing, SonarQube, security scanning (semgrep/gosec/trivy/snyk/gitleaks/nancy), TLA+ model checking.

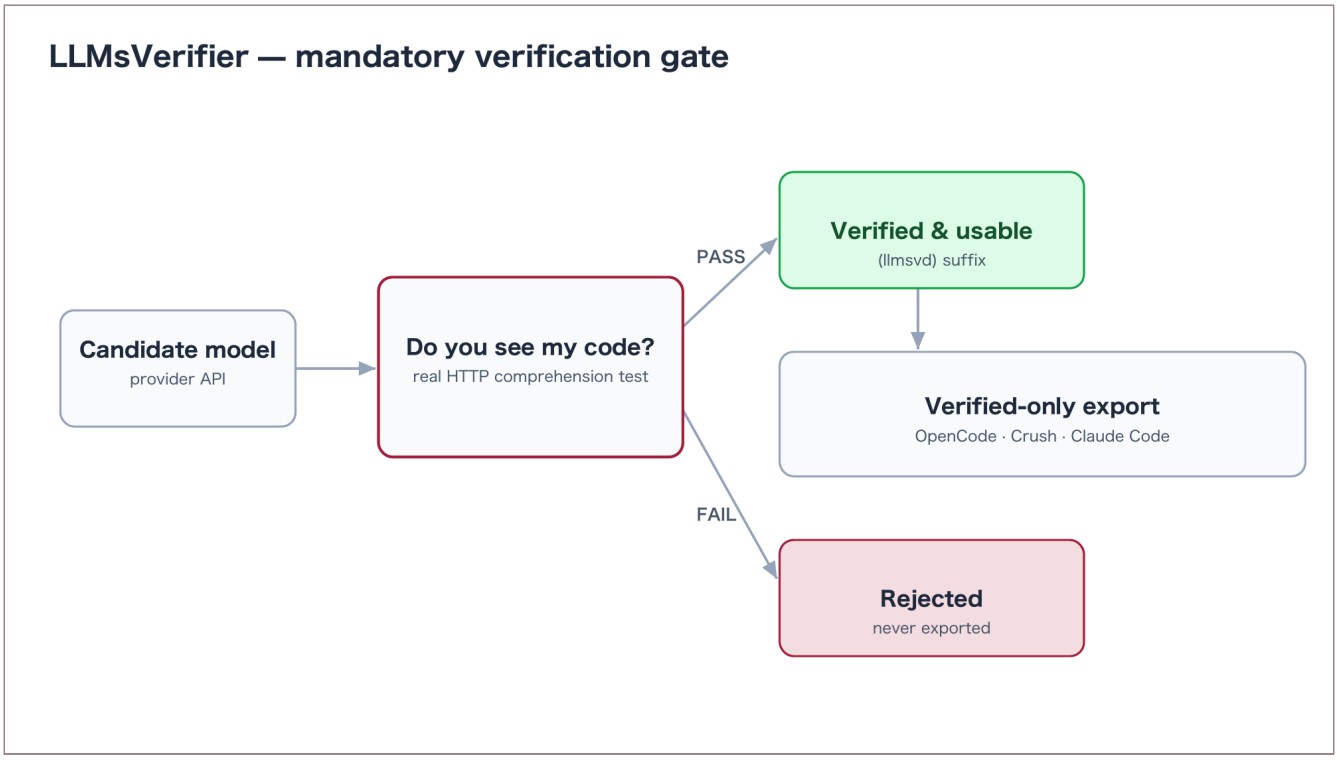
Selected architecture



HelixAgent — an ensemble LLM service where providers debate under a four-stage protocol and ship the answer they agree on, scored by LLMsVerifier.



HelixCluster — a distributed operating system for AI compute, from datacenter GPUs to edge handhelds, under one control plane.



LLMsVerifier — verify, monitor, optimize: the single source of truth for LLM/provider/verification metadata, gated by a mandatory model-comprehension check.