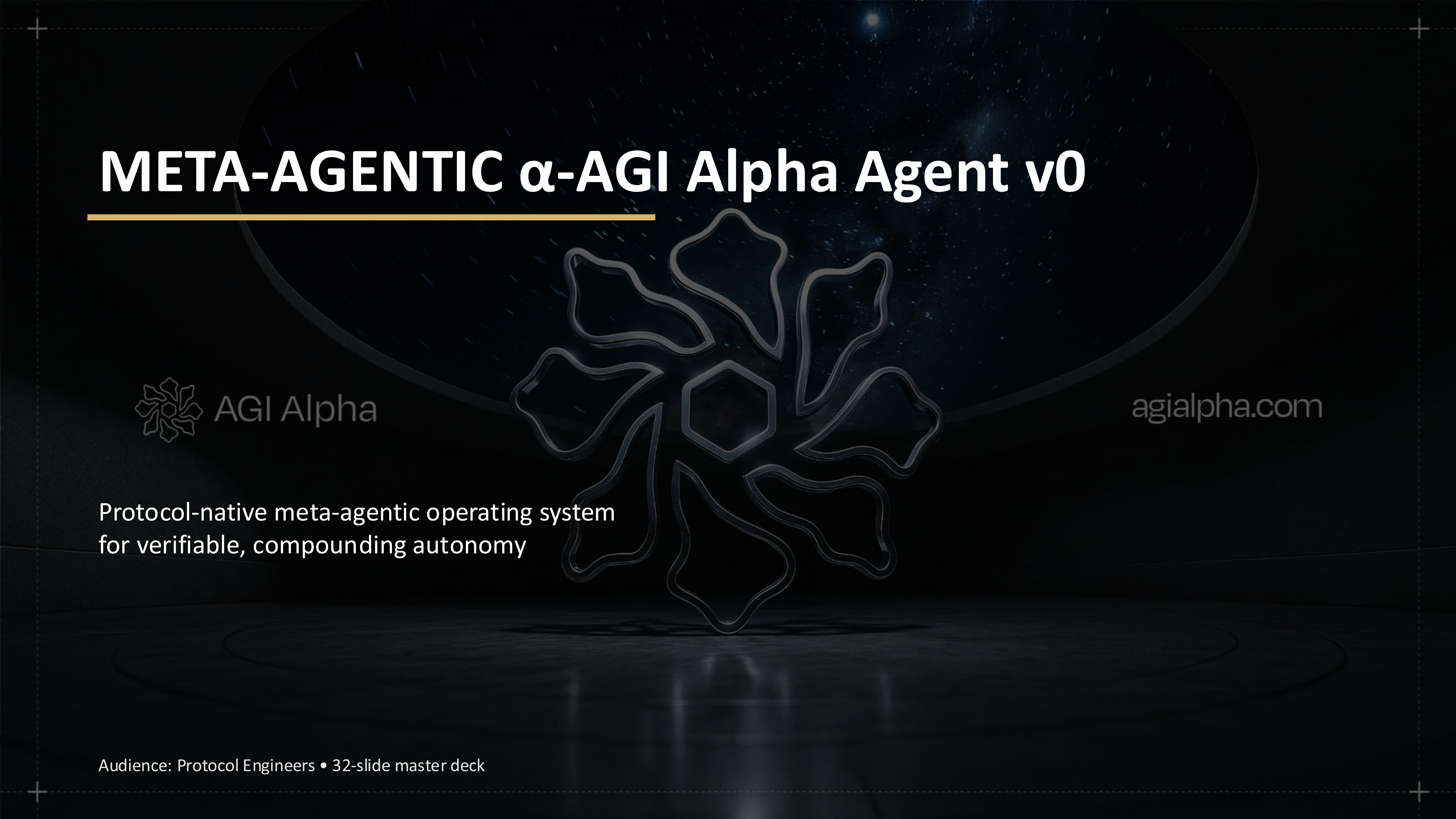
META-AGENTIC α-AGI Alpha Agent v0



Protocol-native meta-agentic operating system
for verifiable, compounding autonomy

Audience: Protocol Engineers • 32-slide master deck

What AGI-Alpha-Agent-v0 is

A meta-agentic coordination core for proof-grade autonomous work.

Identity-bound Autonomy

Every action is attributable to a verifiable actor.
Role-scoped permissions; no ambient authority.

Evidence-first Execution

Deterministic pipelines; replayable proofs.
Artifacts + hashes + logs as a single bundle.

Governed Improvement

Search-driven self-refinement (MATS).
Upgrades gated by tests + policy.

Design invariant: autonomy scales only when the system can prove what it did.

Trust is the throughput limiter

In protocol terms: without verifiability, autonomy cannot clear consensus.

- Opaque agent outputs break reproducibility and audit.
- Unbounded tool access collapses safety and governance.
- No canonical “work” unit → mispricing and incentive drift.
- Fragmented memory → repeated failures, no compounding.

Institutional acceptance criteria

- Tamper-evident logs (hash-chained).
- Replayable execution + deterministic artifacts.
- Role-based controls + policy gates.
- Validation before settlement / promotion.

The meta-agentic loop

Plan → Execute → Prove → Validate → Remember → Improve

Plan
Execute
Prove
Validate
Remember
Improve

Alpha Agent v0 (Brain)

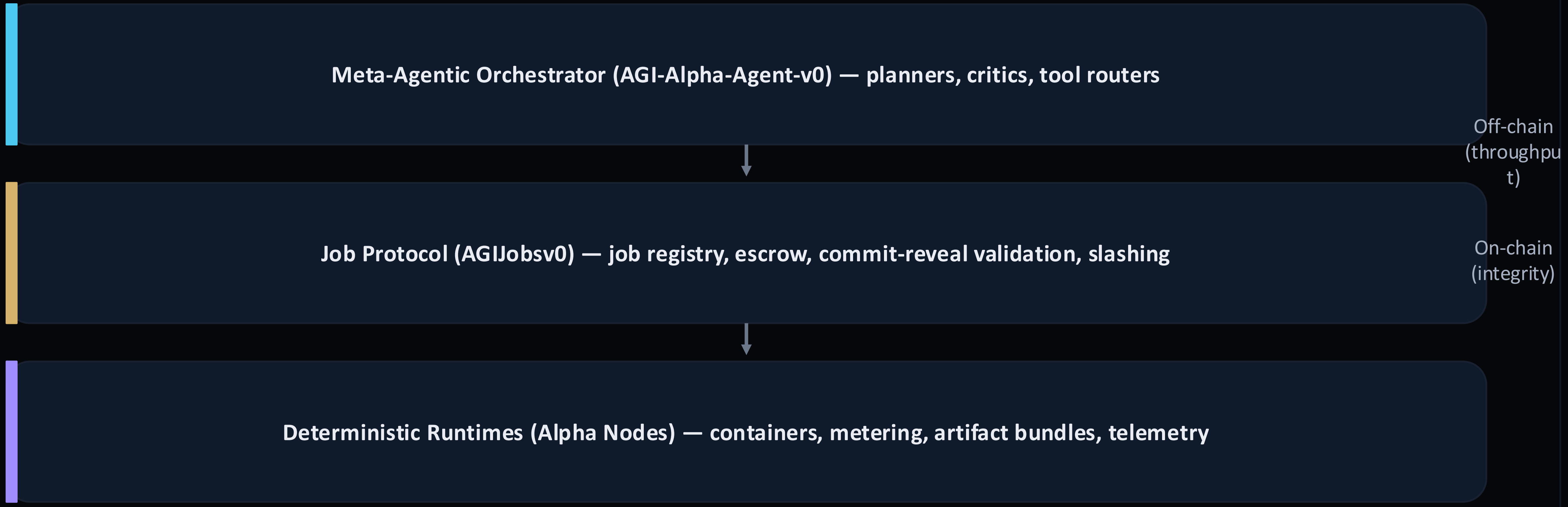
Searches decision space (MATS) under policy.
Decomposes objectives into verifiable jobs.
Attaches evidence requirements to every step.
Promotes only validated artifacts into memory.

Execution fabric (Hands)

AGIJobsv0: job lifecycle + escrow + validation.
Alpha Nodes: deterministic runtimes + proofs.
Chronicle: content-addressed, queryable memory.

Reference architecture (engineer view)

Agent OS on top of a verifiable work + settlement substrate.


Meta-Agentic Orchestrator (AGI-Alpha-Agent-v0) — planners, critics, tool routers
Job Protocol (AGIJobsv0) — job registry, escrow, commit-reveal validation, slashing
Deterministic Runtimes (Alpha Nodes) — containers, metering, artifact bundles, telemetry
Off-chain (throughpu t)
On-chain (integrity)

Contract-verified invariants anchor the system; compute stays off-chain but never unaccounted.

Invariant-driven design

Engineer the system like a conserved-quantity model: what cannot be violated.

No action without attribution

Every act is signed by an identity key.
Role gates prevent privilege drift.

No output without evidence

Artifacts are content-addressed + replayable.
Proof bundles are the unit of truth.

No settlement without validation

Quorum attestation (commit-reveal) before promotion.
Disputes are first-class.

Why it works (physics metaphor)

- Evidence reduces entropy: unverifiable claims are rejected.
- Policies shape the state space (constraints on the Hamiltonian).
- Validation collapses uncertainty into canonical state updates.

MATS: Meta-Agentic Tree Search

Best-first search over candidate plans, with critic-guided branching.

- Node = (state, plan, evidence requirements).
- Expansion proposes edits / patches / new sub-plans.
- Scoring uses validators, tests, and policy compliance.
- Select the frontier node that maximizes expected verified value.

Pseudo-code

```
# Sketch (from DESIGN.md concept)
frontier = PriorityQueue()
frontier.push(root, score=0)

while frontier:
  node = frontier.pop_best()
  if node.is_solution():
    return node
  for edit in propose_edits(node):
    child = apply(edit, node)
    child.score = evaluate(child)  # tests + policy + validators
    frontier.push(child, child.score)
```

Interpretation: a protocol engineer’s “consensus” over plans — only the best, provable branches survive.

Core runtime loop

From objective → job graph → proofs → memory update.

1) Ingest objective + constraints

▼

2) Decompose → DAG of jobs

▼

3) Attach acceptance tests + evidence schema

▼

4) Dispatch to AGIJobsv0 + Nodes

▼

5) Collect proof bundles + validator attestations

▼

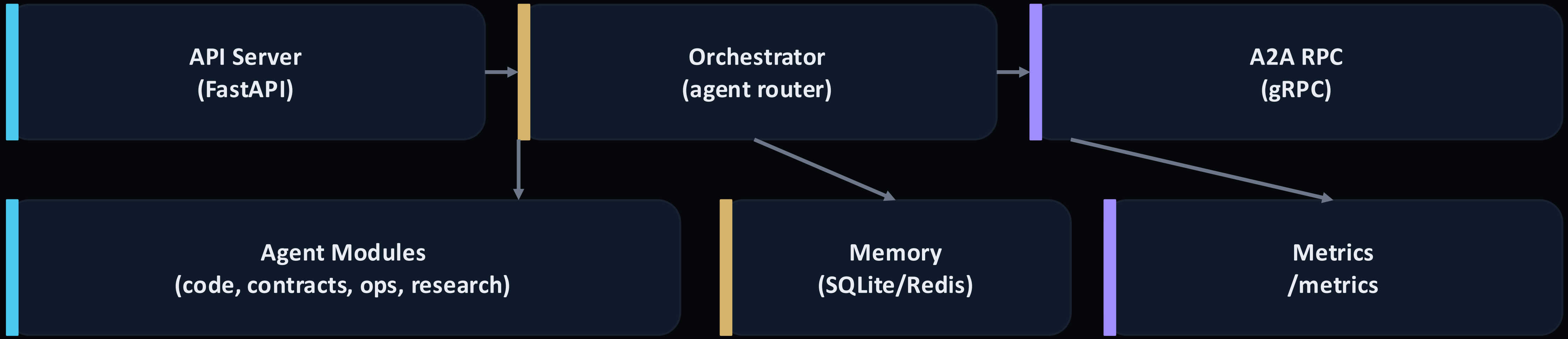
6) Promote validated artifacts into Chronicle

First-class objects (wire format)

- ObjectiveSpec
- PlanGraph (DAG)
- JobSpec + AcceptanceTests
- EvidenceBundleManifest
- ValidatorAttestation
- CanonUpdate / ChronicleEntry

Alpha Factory v1 (reference implementation)

FastAPI control plane + agent suite + memory + metrics.



Key properties: token-secured API, deterministic job artifacts, observability hooks, plugin-style agents.

Operational knobs (env)

ALPHA_ENABLED_AGENTS, ALPHA_CYCLE_SECONDS
ALPHA_REGRESSION_WINDOW / THRESHOLD
API_TOKEN, METRICS_PORT, A2A_PORT

Dependencies

FastAPI + Uvicorn
SQLite/Redis (memory)
Prometheus / OpenTelemetry-style metrics

REST API surface (Alpha Factory)

A minimal, scriptable control plane for engineers.

Endpoints

```
GET  /healthz
GET  /agents
POST /agent/{name}/trigger
POST /agent/{name}/update_model
POST /agent/{name}/skill_test
POST /memory/append
GET  /memory/query?q=...
GET  /metrics
```

Security posture

Bearer token required for stateful ops.
Separate metrics port recommended.
Disable TLS only for local dev.

Engineer workflows

Trigger agents from CI jobs / bots.
Run skill tests as admission control.
Query memory for deterministic context.

A2A wire envelope

A minimal protobuf envelope for agent-to-agent RPC.

alpha_factory_v1/core/utils/a2a.proto

```
syntax = "proto3";

message Envelope {
  string sender = 1;
  string recipient = 2;
  string msg_type = 3;
  bytes payload = 4;
  int64 timestamp = 5;
}
```

Design intent

Keep transport generic; payload carries domain schema.
Sender/recipient are identity-bound names.
Timestamp enables ordering + audit trails.

Recommended extension pattern

Define payload protobufs per capability (e.g., JobSpec, EvidenceBundle, Attestation).
Hash + sign payloads; include signature metadata either in payload or an outer signed wrapper.

Evidence bundle as a unit of truth

Protocol engineers: treat outputs like blocks — content-addressed, replayable, signed.

Example: EvidenceBundleManifest

```
{
  "jobId": "0x…",
  "inputs": { "specHash": "0x…", "dataURI": "ipfs://…" },
  "runtime": { "image": "sha256:…", "seed": 1337 },
  "outputs": { "resultURI": "ipfs://…", "resultHash": "0x…" },
  "logs": { "stdout": "ipfs://…", "traceHash": "0x…" },
  "signatures": ["0xnodeSig", "0xvalidatorSig…"]
}
```

- Determinism: pinned image + fixed seed.
- Content addressing: hashes bind every artifact.
- Replay: validators reproduce resultHash.
- Admissibility: signatures establish provenance.

Determinism stack

Make compute reproducible so verification is cheap and definitive.

Execution

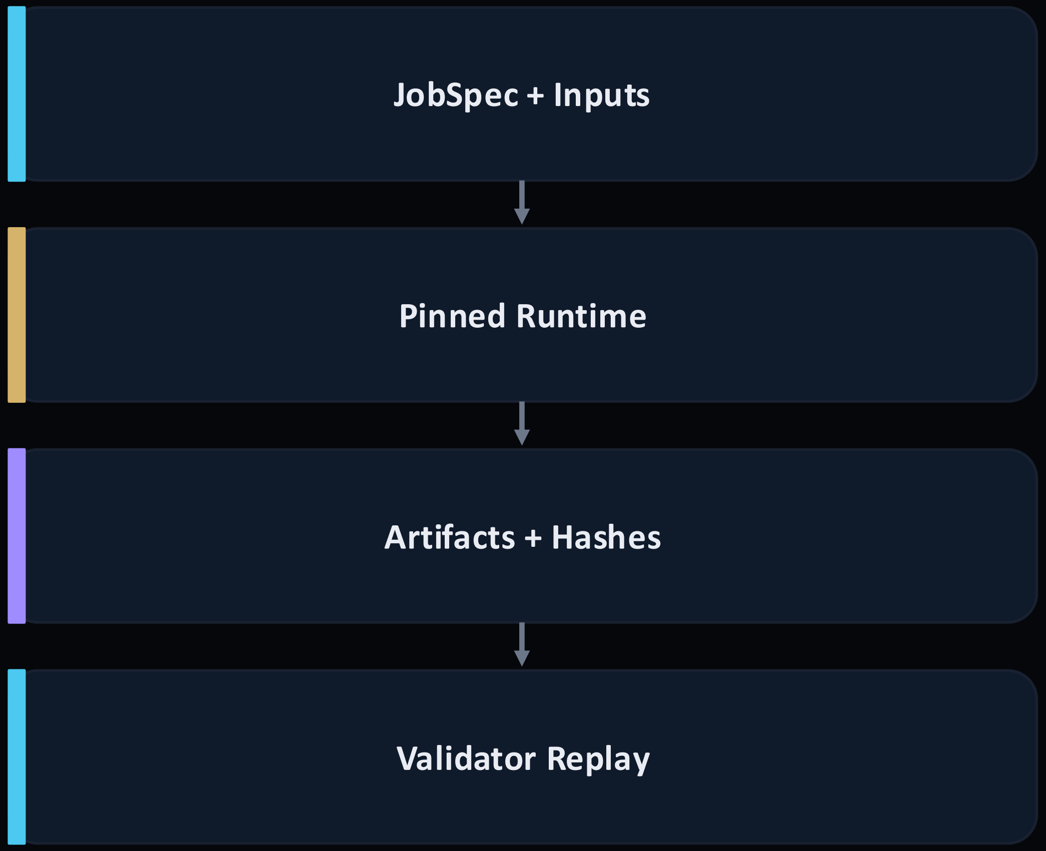
Container images pinned by digest.
Fixed seeds; captured env + deps.
Network egress policy: default deny.

Instrumentation

Sidecars capture logs + metrics.
Hash chaining for traces.
Resource metering (e.g., compute-seconds).

Verification

Replay locally or via validators.
Consensus over resultHash.



Autonomy under authority

Policy is the execution boundary — enforced both off-chain and on-chain.

- Tool allowlists (models, endpoints, RPC methods).
- Budget caps and timeouts per job.
- Data residency and redaction constraints.
- Circuit breakers on anomaly signals.

Where enforcement lives

Node runtime sandbox (hard stop).
Agent router (soft policy + routing).
Job contracts (escrow + validation gates).

Failure containment

Non-compliant jobs are rejected pre-execute.
Quarantine modes for suspicious job classes.
Operator override remains final.

Separation of duties (agents as roles)

Game-theoretic robustness through independent incentives and checks.

Generator

Proposes plans, patches, jobs.
Maximizes progress under constraints.

Critic

Finds failure modes.
Forces better evidence and tests.

Validator

Replays + attests.
Quorum finalizes truth.

Sentinel / Monitor

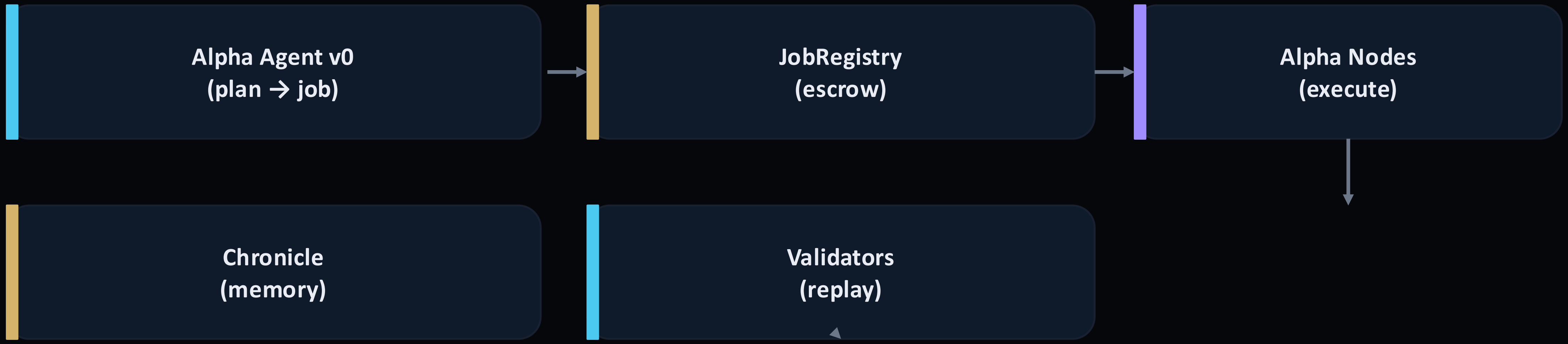
Observes anomalies (telemetry, drift, policy breaches).
Trips circuit breakers; escalates disputes.

Governor

Owns policy + budgets + upgrades.
Uses logs as institutional accountability.

Integration with AGIJobsv0

Alpha Agent v0 turns plans into jobs that can be escrowed, verified, and settled.



Practical outcome

Your agent outputs become protocol objects: jobs, proofs, attestations, and canonical memory entries.
The system’s “intelligence” is the compounding set of validated artifacts — not opaque tokens.

On-chain touchpoints (JobRegistry excerpt)

Engineers: the minimal callable surface Alpha Agent needs.

Solidity signatures

```
// contracts/v2/JobRegistry.sol (excerpt)
function createJob(string calldata uri, uint256
reward) external returns (uint256 jobId);
function apply(uint256 jobId, uint256 bond)
external;
function submit(uint256 jobId, string calldata
resultURI, bytes32 resultHash) external;
function dispute(uint256 jobId, string calldata
reasonURI) external;
function finalize(uint256 jobId) external;
```

Engineering notes

URI points to JobSpec (IPFS/Arweave) with acceptance tests + evidence schema.
resultHash commits the exact output promoted into Chronicle.
dispute() escalates; finalize() releases escrow after validator quorum.

JSON ABI fragment (ethers-ready)

```
[
  {"type":"function","name":"createJob",
   "inputs":[{"name":"uri","type":"string"},{"name":"reward","type":"uint256"}],"outputs":[{"name":"jobId","type":"uint256"}]},
  {"type":"function","name":"finalize","inputs":[{"name":"jobId","type":"uint256"}],"outputs":[]}
]
```

Validation modes

How truth is produced: deterministic replay + quorum.

- Fast path: deterministic replay produces identical resultHash.
- Commit–reveal prevents herding / influence.
- Quorum thresholds are policy-set per job class.
- Disputes trigger deeper review or additional validators.

Quality signals

Acceptance tests + property checks.
SLO adherence (latency, uptime).
Resource metering consistency.

Anti-Byzantine incentives

Stake-weighted selection.
Slashing for dishonest attestations.
Reward honest minority that catches faults.

Chronicle: decision-relevant memory

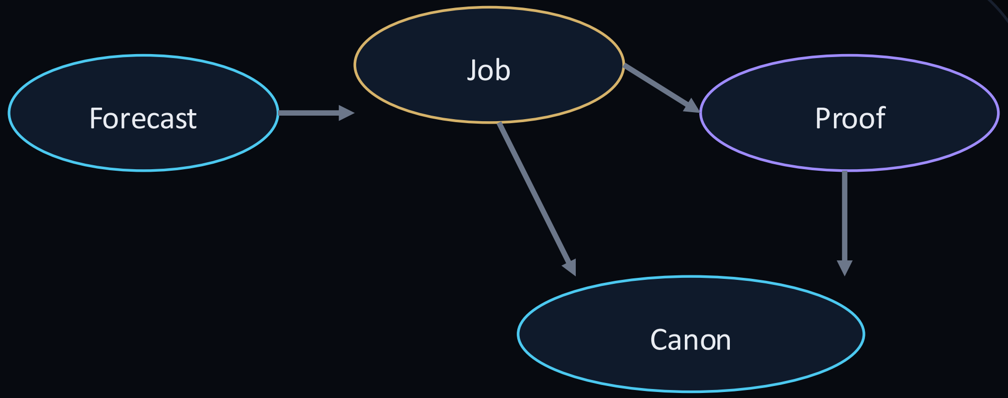
A queryable graph of objectives → jobs → proofs → outcomes.

First-class entities

Objectives, Plans, Jobs, Agents, Nodes
Artifacts, Proof Bundles, Attestations
Policies, Upgrades, Settlement Events

Why it compounds

Validated outputs become reusable primitives.
Failures are retained as anti-patterns.
Meta-agent learns on proven data only.



Traceability is a graph traversal — not a narrative.

Observability and SLOs

Telemetry is part of evidence; metrics are signed and audit-ready.

Signals

Traces: objective → job → proof chain
Metrics: latency, success %, resource usage
Logs: hash-chained runtime + agent logs

Controls

SLO gates (deadline, uptime)
Anomaly detectors (drift, outliers)
Circuit breakers (pause / quarantine)

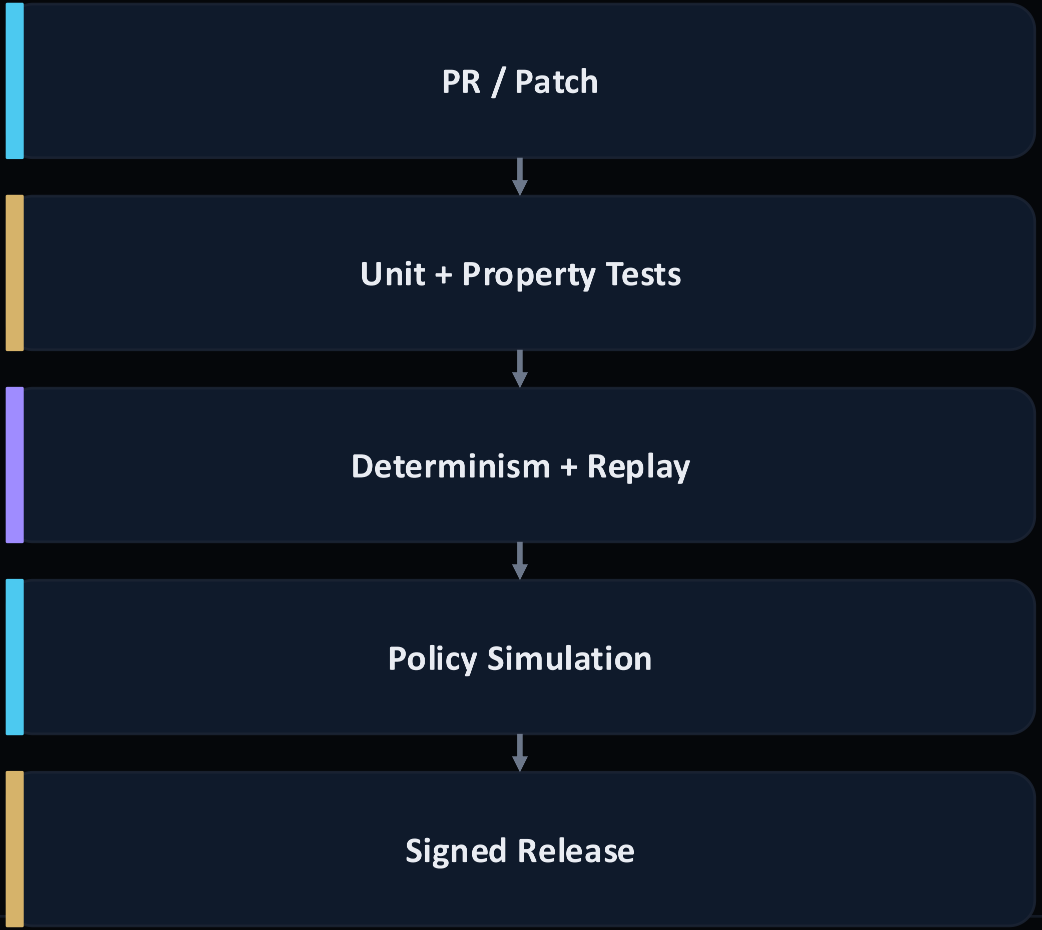
Example metrics

```
# /metrics (Prometheus format)
alpha_jobs_total{status="validated"} 128
alpha_jobs_latency_p95_seconds 4.2
alpha_evidence_replay_mismatch_total 0
alpha_policy_denied_total 3
```

CI as protocol: “main is deployable truth”

A meta-agent is only as safe as its upgrade pipeline.

- Deterministic tests + regression thresholds are first-class.
- Skill tests gate new agent capabilities.
- Policy simulations run before deployment.
- Artifacts (builds, SBOMs, hashes) are stored and referenced.



Deployment modes

Start local, scale to clusters, federate across sovereign domains.

Local Dev

Single process
SQLite memory
Mocked tools

Single Node

Token-secured API
Redis optional
Metrics enabled

Kubernetes

Horizontal scaling
Secret management
Ingress + TLS

Federated / multi-tenant (institutional)

Separate trust domains; shared job protocol.
Namespaced policies; isolated runtimes.
Cross-domain attestation via signed evidence bundles.

Security model

Keys, boundaries, and least privilege — engineered, not assumed.

- Key separation: signing vs on-chain execution vs API auth.
- HSM / multisig for high-impact actions (upgrades, treasury).
- Secrets in env only for dev; vault/KMS in prod.
- Default-deny network policy for job containers.

Attack surface reductions

Determinism: replay catches tampering.
Policy gates: prevent exfiltration paths.
Validator redundancy: defeats single-node fraud.

Operational hardening

Rate limits + circuit breakers.
Audit log retention + SBOMs.
Continuous dependency scanning.

Threat model (protocol lens)

Assume Byzantine actors; engineer incentives and verification.

Adversaries

Malicious worker (fake results).
Byzantine validator cartel.
Prompt/tool injection into agent router.
Data exfiltration via side channels.

Mitigations

Replay + quorum; stake-slashing.
Commit–reveal; randomized selection.
Tool allowlists; sandbox; egress control.
Signed logs; anomaly detection; redaction.

Game-theoretic invariant

```
Goal: make honesty the dominant strategy

Expected payoff(cheat) < Expected payoff(honest)

by: slashing + low probability of undetected fraud.
```

Kardashev-II framing: cognitive energy

Engineer for planetary compute without losing control of state.

Energy analogy

Compute ≈ available free energy (capacity).
Policies constrain feasible trajectories.
Proofs reduce entropy in decision outputs.

Scaling principle

Throughput scales with nodes; integrity scales with validation.
Chronicle ensures past work is reusable (compounding).
Meta-agent optimizes allocation under constraints (like a control system).

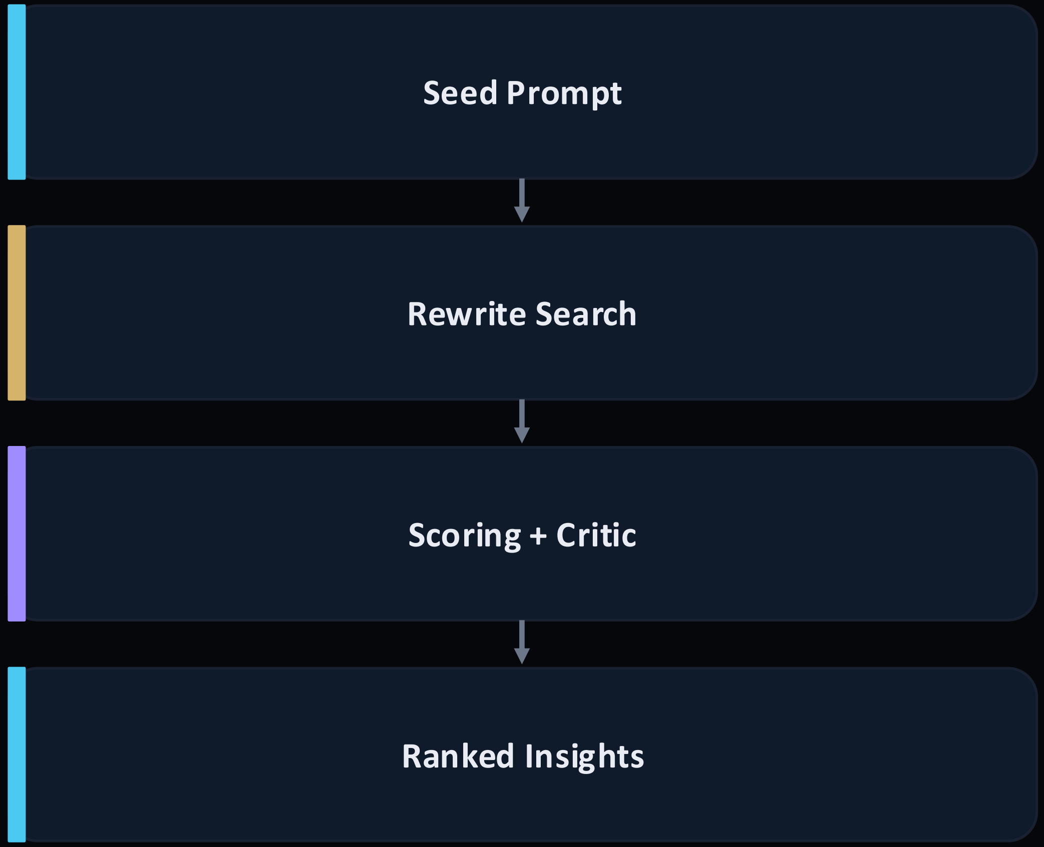
State update rule

```
Let S be system state (Chronicle)

ΔS = f(verified_work)
verified_work = Σ_i 1[replay_ok ∧ policy_ok ∧ tests_ok]

Only verified work updates S → stable long-horizon behavior.
```

Example: α-AGI Insight (offline demo)

Best-first search over rewrite chains to surface actionable hypotheses.

- Inputs: prompt + optional dataset.
- Process: generate candidate rewrites; score; expand best.
- Output: ranked "insights" with evidence pointers.
- Constraint: offline-friendly; deterministic scoring hooks.



Example: patch-to-proof pipeline

Generate → test → replay → attest — then promote.

- 1) Agent proposes patch
- 2) Run unit tests in pinned container
- 3) Produce artifact bundle + hashes
- 4) Validators replay
- 5) Merge / promote only if verified

Artifact discipline

```
git diff -- patch

+ if (policy.ok && tests.pass) {
+   evidence = bundle(outputs, logs, hashes)
+   submit(jobId, evidence)
+ }
```

Why engineers care

Patch provenance becomes a verifiable object.
Replays catch nondeterminism or hidden dependencies.
Governance can require stronger validators for high-risk changes.

Governance & upgrade discipline

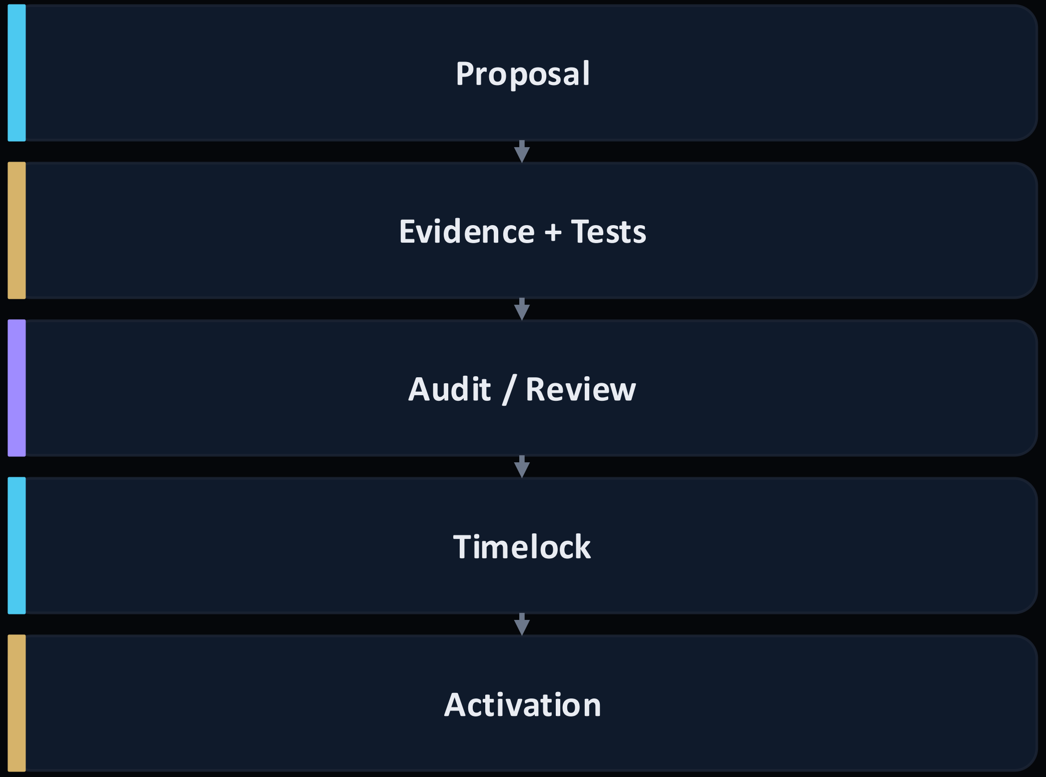
Power without oversight is failure; upgrades are evidence-gated.

Governable objects

Policies (tool allowlists, budgets)
Validation thresholds per job class
Approved runtime images / SBOMs

Controls

Timelocks + multisig for upgrades
Emergency stop (global pause)
Versioned canon definitions



Interop and extensibility

A meta-agent is a routing layer — plugins are capability modules.

- Agent modules can wrap external systems (RPC, APIs, toolchains).
- Policy determines which tools are callable in which contexts.
- Evidence schemas let validators reproduce tool calls (or verify outputs).
- A2A envelope supports multi-agent mesh topologies.

Plugin contract (recommended)

Input schema + deterministic config
Acceptance tests
Evidence manifest spec
Replay strategy (full/partial)

Engineer win

Swap implementations without changing invariants.
Incremental adoption: start off-chain, anchor later.

Roadmap (engineer-credible)

Scale capability only as fast as proof + governance scales.

v0

Reference agent OS
Offline demos
Basic proof bundles

v1

Tighter job protocol bindings
Validator automation
Better memory indexing

v2

Policy DSL + simulators
Federation primitives
Formal verification hooks

North star

A machine that can plan, act, and improve — while remaining fully auditable and governable.
Compounding capability emerges from the Chronicle of validated artifacts.

Repository orientation

Where protocol engineers typically start reading.

Top-level map

```
AGI-Alpha-Agent-v0
├─ docs/ (OVERVIEW, ARCHITECTURE, DESIGN)
├─ alpha_factory_v1/
│   ├─ backend/ (FastAPI, agent router, metrics)
│   └─ core/ (A2A envelope, utilities)
├─ contracts/ (reference on-chain components)
└─ demos/ (offline and end-to-end examples)
```

Suggested reading path

1) docs/OVERVIEW.md
2) docs/DESIGN.md (MATS)
3) alpha_factory_v1/backend/api_server.py
4) contracts/v2/JobRegistry.sol (integration)

Contribution lanes

Determinism + replay tooling
Policy DSL + simulators
Validator automation
Better schemas + docs

Build the machine that proves its work

The fastest path to safe autonomy is verifiable autonomy.

AGI-Alpha-Agent-v0 operationalizes a simple idea:

When intelligence becomes infrastructure, every decision must be replayable, every claim must be provable, and every upgrade must be governable.

Repo: github.com/MontrealAI/AGI-Alpha-Agent-v0

Companion protocols: AGIJobsv0 • AGI-Alpha-Node-v0