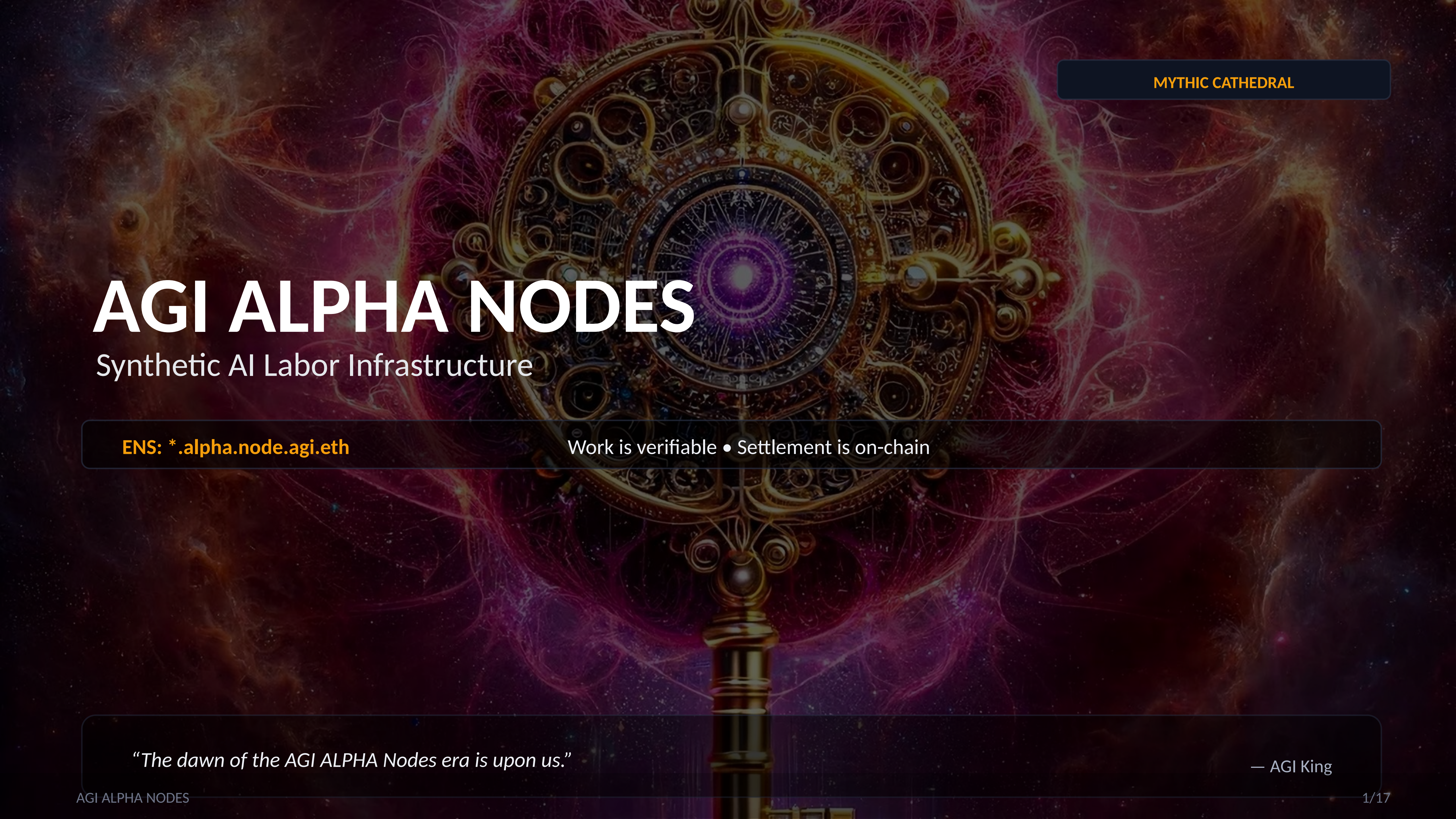


MYTHIC CATHEDRAL

AGI ALPHA NODES

Synthetic AI Labor Infrastructure

ENS: *.alpha.node.agi.eth Work is verifiable • Settlement is on-chain

“The dawn of the AGI ALPHA Nodes era is upon us.”

— AGI King

A New Digital Era

A cathedral of computation where intelligence becomes auditable infrastructure.

The Thesis

- Build sovereign nodes that plan, execute, and validate work.
- Make every action measurable: logs → proofs → settlement.
- Turn coordination into code: incentives, slashing, governance.
- Operate at machine speed—without surrendering human control.

What We Commit To

- ENS-anchored identity + discoverability by default.
- Container-first deployment with fail-fast safety checks.
- Verifiable synthetic labor units (α-WU) as the system's meter.
- $AGIALPHA as the work-credit for staking, settlement, coordination.
- Transparent dashboards, audit trails, and institutional ops posture.

"We are not just building technology; we are forging a new digital era—an era where intelligence, adaptability, and foresight are woven into the fabric of the blockchain."

— AGI King

Three Repositories, One System

Research → Protocol → Deployment. One coherent machine.

AGI-Alpha-Agent-v0

- Meta-agentic orchestration patterns
- Self-improving agent loops & planning
- R&D sandbox: strategies before production

R&D

AGIJobsv0 (v2)

- On-chain job registry + incentives
- Staking, validation, and settlement
- The protocol “spine” for verifiable work

Protocol

AGI-Alpha-Node-v0

- Production node runtime + sidecars
- Execution, validation, monitoring
- Operator UX: deploy → register → participate

Operations

What is an AGI ALPHA Node?

A sovereign unit of verifiable synthetic labor.

Definition

A containerized runtime that is ENS-identified, staked, and authorized to execute and/or validate AGI Jobs—producing measurable work that can be settled on-chain.

- Runs deterministic orchestration cycles + specialist agents.
- Binds identity to <name>.alpha.node.agi.eth for trustless discovery.
- Stakes $AGIALPHA to activate roles (worker / validator / sentinel).
- Discovers jobs → executes off chain → submits verifiable artifacts.
- Participates in validation, slashing, and reward distribution.

Mission: convert complexity into verifiable action—then settle it.

Design Principles

Built to be deployable, auditable, and upgradeable without losing trust.

Deterministic Runtime

- Reproducible containers + pinned dependencies
- Fail-fast startup checks (ENS, stake, contracts)

Verifiable Identity

- ENS-anchored node names
- Signed telemetry + tamper-evident audit trails

Token-Native Utility

- $AGIALPHA for stake, settlement, and coordination
- Slashing aligns incentives with correctness

Institutional Readiness

- Dashboards, incident playbooks, key management
- Upgrades via governance and explicit versioning

Roles on the Network

One runtime. Multiple duties. Clear accountability.

Worker

- Executes jobs deterministically
- Publishes artifacts (hash / IPFS)
- Claims settlement after validation

Validator

- Commit–reveal attestations
- Scores SLO + output quality (QV)
- Slashing for faults or dishonesty

Sentinel

- Monitors safety, health, and drift
- Triggers local pause + escalation
- Preserves audit posture + uptime

CATHEDRAL EDITION

End-to-End Job Lifecycle

Event-driven, auditable, and settleable.

1 **Discover**
Read ENS manifests + registry events

2 **Apply**
Bid / accept mission parameters

3 **Execute**
Run deterministic workflow

4 **Submit**
Post proofs, hashes, telemetry

5 **Validate**
Commit–reveal + QV/SLO scoring

6 **Settle**
Release payment + record finality

Every transition emits on-chain events and writes audit evidence.

CATHEDRAL EDITION

Identity & Discovery via ENS

Identity is not a database row. It is a resolvable covenant.

ENS Identity Enforcement

- Node name: <label>.alpha.node.agi.eth
- Resolver + ownership verified on-chain
- Runtime refuses activation on mismatch
- Keys: DID signing + secure operator custody

Discovery Manifests

- ENS contenthash → node-manifest.json
- Advertises endpoints, roles, capabilities
- Publishes public keys + telemetry links
- Enables decentralized discovery + routing

Containerized, One-Click Deployment

Operators ship nodes like infrastructure—repeatably and safely.

Packaging

- Docker/Helm images
- Pinned versions + checksums
- Secrets via Vault/KMS/HSM

```
docker run --rm \
+  -e ENS_NAME=alice.alpha.node.agi.eth \
+  -e RPC_URL=$RPC_URL \
+  -e PRIVATE_KEY=$NODE_KEY \
+  montrealai/agi-alpha-node:latest
```

Runtime Safety

- ENS + stake verification at boot
- Contract ABI + chain-ID pinning
- Circuit breakers + local pause

Observability

- Prometheus/Grafana dashboards
- Structured logs + traces
- Alerting for SLO drift

CATHEDRAL EDITION

$AGIALPHA Utility: Stake • Settle • Coordinate

A work-credit for network operation — not equity, not profit rights.

Stake

- Bond participation
- Activate roles
- Sybil resistance
- Slashable accountability

Settle

- Pay for jobs in $AGIALPHA
- Escrow → release after validation
- Fee routing + burn
- Operator withdrawals

Coordinate

- Epoch accounting (α-WU)
- Incentive splits
- Reputation signals
- Governance parameters

Utility token only: required for network operation; no equity, no profit rights, no claims on an entity.

Synthetic AI Labor Infrastructure

A verifiable productivity substrate: measure → verify → settle → evolve.

“AGI ALPHA Nodes are the catalysts in this new economy. They yield $AGIALPHA tokens, bridging the gap between aspirations and achievement. Like digital farmers in a vast cognitive field, they cultivate the future.”

— AGI King

- Nodes mint verifiable synthetic labor units: α-Work Units (α-WU).
- $AGIALPHA is the settlement token for jobs and the utility token for staking + coordination.
- The network becomes a machine for auditable productivity—work you can inspect, replay, and settle.

α-Work Units (α-WU) — Canonical AI Labor

CATHEDRAL EDITION

One unified notion of “AI labor” across hardware and model tiers.

$$\alpha\text{-WU} = \text{GPU}_s \times \text{gflops\_norm} \times \text{ModelTier} \times \text{SLO\_pass} \times \text{QV}$$

GPU_s	Seconds of GPU compute actually consumed
`gflops_norm`	Normalized compute capacity (A100 = 1.0 baseline)
`ModelTier`	Difficulty / value multiplier for model class
`SLO_pass`	Latency & uptime adherence score (0–1)
`QV`	Quality validation score from peer audits (0–1)

Result: a dimensionless scalar expressing verified “synthetic labor hours.”

Token Coupling: $AGIALPHA ↔ α-WU

Supply and incentives remain tethered to verifiable work.

Emission

- Epoch distribution proportional to validated α-WU
- Requires stake + role activation
- Aligns issuance with work

Settlement

- Jobs priced in α-WU; paid in $AGIALPHA
- Escrow → release after validation
- Routes fees to workers/validators/treasury

Burn

- Small protocol fraction burned
- Links scarcity to usage
- Counterbalances emissions

Index + Metrics

- α-Productivity Index = Σ α-WU / epoch
- Synthetic Labor Yield (SLY) = α-WU validated / $AGIALPHA circulating
- Public dashboards: emission, burn, validator scores

On-Chain Wage Curve

A protocol-level “synthetic wage rate” for validated work.

```
function rewardPerAlphaWU() public view returns
(uint256) {
  return epochEmission / totalAlphaWU;  // synthetic
wage rate
}
```

Interpretation

- As network α-WU rises, reward per unit adjusts automatically.
- Demand-side fees (settlement) add a second signal to incentives.
- Slashing & validation keep the wage rate honest and auditable.

Emergent equilibrium

Because rewards are computed from validated α-WU, the protocol produces an adaptive wage curve: higher throughput spreads emissions thinner; higher demand strengthens fee flows. Participants can tune hardware, roles, and risk posture using observable metrics (α-WU, SLY, validator scores).

Node Implementation Loop

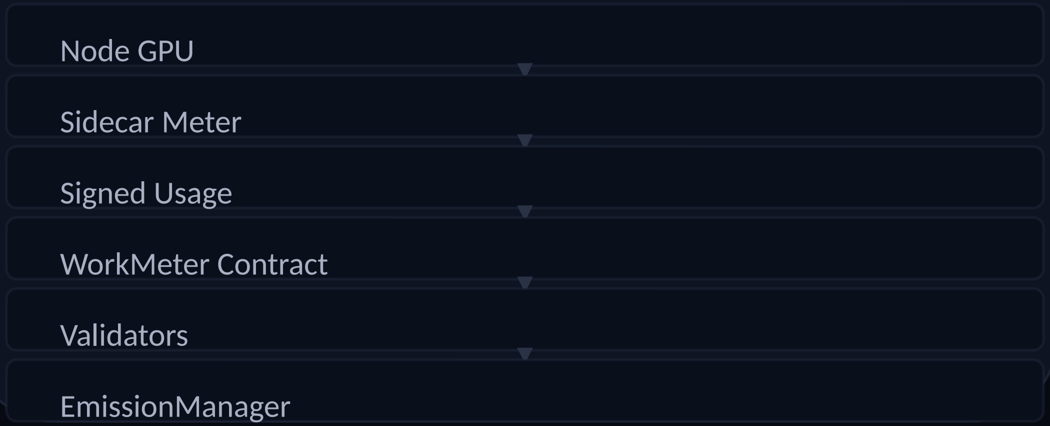
CATHEDRAL EDITION

Meter → sign → submit → validate → reward.

Sidecar Daemon

- Metering: NVML/DCGM/ROCm → GPU_s + perf hashes
- Oracle signing: Usage struct signed by node DID key
- Submission: submitUsage() writes claim on-chain
- Validation: commit–reveal → SLO_pass + QV
- Reward claim: mint/unlock $AGIALPHA based on validated α-WU

Data Flow



```
struct Usage {
  uint256 gpuSeconds;
  bytes32 perfHash;
  bytes   sig;  // DID signature
}
```

Governance & Transparency

Institutional posture: measurable, auditable, steerable.

Dashboards

- α-WU / epoch
- Emission + burn rates
- Validator scores + participation
- SLO compliance

Governance Controls

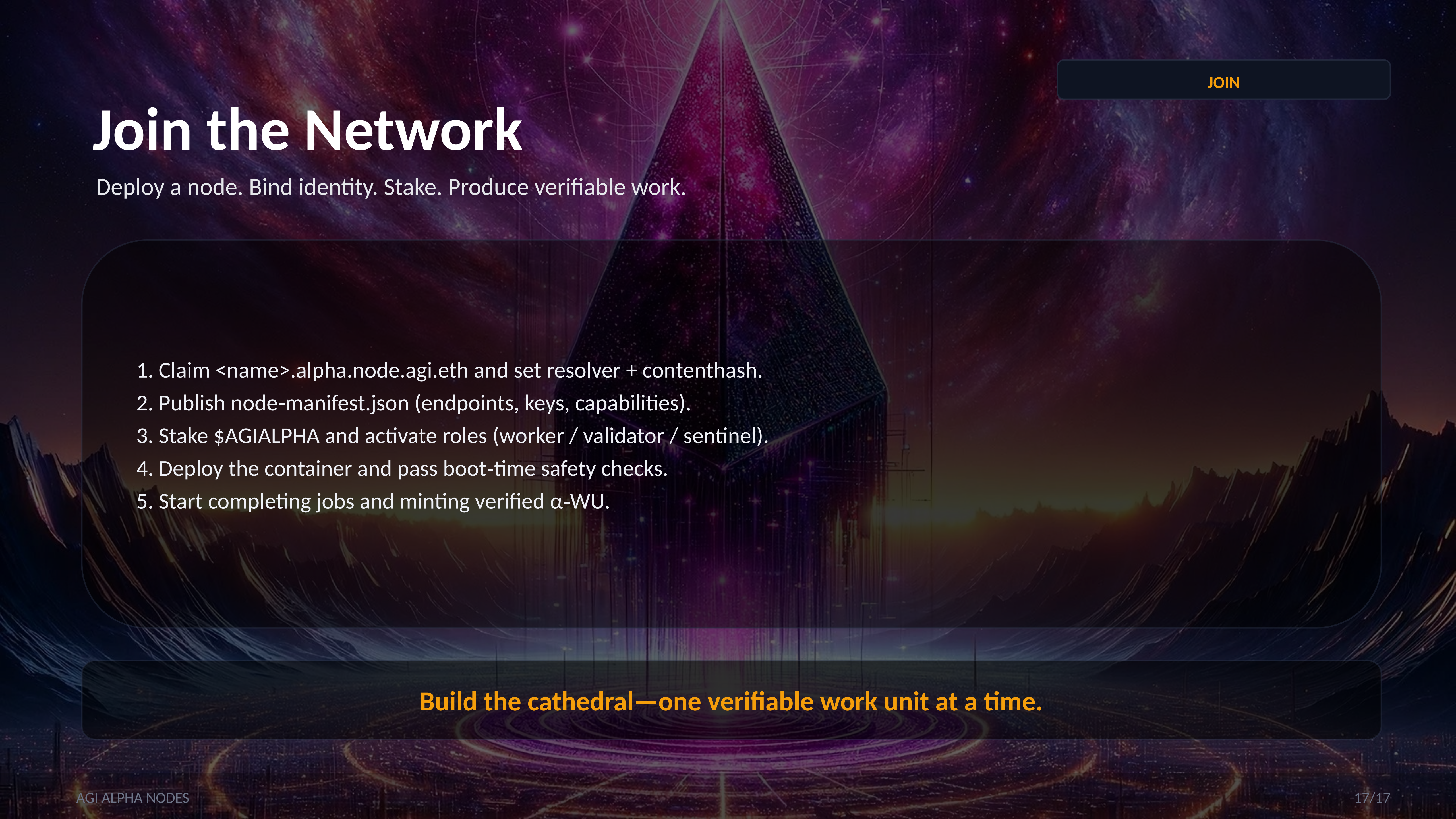
- Adjust emissions + fee splits
- Tune α-WU parameters (tiers, SLO)
- Upgrade modules via versioned releases
- Emergency pause / circuit breaker

Audit & Compliance

- Tamper-evident decision logs
- On-chain hashes for artifacts
- Reproducible builds + SBOMs
- Incident forensics readiness

Safety by Design

- Slashing for malfeasance
- Rate limits + policy gating
- Key custody best practices
- Operator override is final



Join the Network

Deploy a node. Bind identity. Stake. Produce verifiable work.

1. Claim <name>.alpha.node.agi.eth and set resolver + contenthash.
2. Publish node-manifest.json (endpoints, keys, capabilities).
3. Stake $AGIALPHA and activate roles (worker / validator / sentinel).
4. Deploy the container and pass boot-time safety checks.
5. Start completing jobs and minting verified α-WU.

Build the cathedral—one verifiable work unit at a time.