FUTON0

@multiarg futon0/devmap
@title FUTON0 Development Map
@audience futon-devs, reflective-agents, future-you
@tone formal-analytic
@style roadmap
@allow-new-claims true
@length any
@factor Mindful apprehension (sati + sampajañña)
**@IFR: FUTON0 is the mindfulness and futon-perception layer: the lived,
experiential basis through which real-but-virtual futures (futons) become
intelligible via symbolic mediation. It stabilises vitality, rhythm, clarity,
and the conditions for "wondrous apprehension". FUTON0 provides a
bio-mechanical interface to the extended cognitive architecture—the cybernetic
floor from which the whole FUTON stack perceives, adapts, and co-evolves with
its future.**

@state Vitality indicators exist but are not yet integrated; reflective and
symbolic channels exist but are only loosely coupled; futonic insights arise in
narrative but are not yet expressed in computational terms. FUTON3 is still
awaiting improved devmaps for several layers, which currently blocks clear
narratives about the futons it will eventually describe.

@next Maintain and develop minimal models of vitality signals; cultivate
periodic wondrous apprehension; and continue light futon-prototyping efforts
with steady awareness, so that both clear devmaps and concrete implementations
can emerge naturally.

! instantiated-by: Prototype 0 — System Vitality Indicators [💤/系统]
+context: FUTON0 provides the minimal embodied/creative signals required for early detection of imbalance, while avoiding expansion of lifelogging overhead.
+if: You want dependable early warning of drift in health, energy, or expressive rhythm.
+however: Existing signals are scattered (notes, reminders, implicit habits) and therefore unreliable as a coherent early-warning interface.
+then: Use only *existing traceable activity* as indicators: Tatami sessions, git commits, filesystem timestamps, and micro-recording cadence. Add one lightweight prompt-based behaviour: a weekly tai-chi intention check surfaced automatically by the FUTON3 HUD. No manual lifelogging is introduced.
+because: Indicators remain sustainable only when they create no friction. A minimal interface built from pre-existing behavioural traces provides consistent, low-noise signals that prevent hidden imbalance from migrating up-stack.
+evidence:
- The FUTON0 clock already exists: Tatami/API chat sessions automatically
emit timestamped events, providing a ground truth for daily activity.
- Git commits, recording timestamps, and filesystem modification times supply
ambient movement/engagement signals with zero added logging.
- FUTON1 work-session tags already show >90% end-of-session coverage.
+next: Build the cron-based vitality scanner; implement the weekly tai-chi HUD prompt; run a 30–45 day pilot to stabilise thresholds; and refine the vitality interface only after reviewing correlations with mood, focus, and WIP drift.
+next-evidence:
- Extend the cron scanner to generate a daily summary of filesystem
activity, recording cadence, and Tatami timestamps.
- Capture tai-chi intention Y/N responses in FUTON3, creating a new sparse
evidentiary channel without manual logging.
- Implement a lightweight inference pass (FUTON3 or FUTON4) to interpret
these signals for trend detection and early-warning triggers.

! instantiated-by: Prototype 1 — Epistemic Rhythm & Salients [💖/弓]

FUTON1

@multiarg futon1/devmap
@title FUTON1 Development Map
@audience futon-devs, infra-agents, future-you
@tone formal-analytic
@style roadmap
@allow-new-claims true
@length any
@factor Keen investigation (dhammavicaya)
**@IFR:
FUTON1 provides a deterministic, storage-oriented substrate that maintains a
coherent graph of facts across sessions. It keeps Datascript (fast) and XTDB
(persistent) in reproducible alignment so that the same world state appears
reliably across restarts and environments. It exposes a minimal CLI and HTTP
surface for ingesting text, inspecting entities and relations, and emitting
stable summaries—such as focus headers—that downstream code can depend on
without embedding persistence or orchestration logic.**

FUTON1 supports keen investigation by making the fact graph inspectable and
faithful to stored state. Its role is limited to storage, hydration, and
deterministic summarisation: it does not prescribe inference or policy. When
this layer operates cleanly, other futons inherit a durable and predictable
world model on which more complex behaviours can be built.

@state
The deterministic demo/client workflow, Datascript↔XTDB mirroring, and
open-world ingest pipeline are operational and well-documented, with hydration
and persistence behaving as expected. The NLP interface, graph-memory module,
and configuration surfaces are functional but unevenly documented, and their
tests do not yet cover all intended entry points. The HTTP API provides a
stable interaction surface, though endpoint contracts and data shapes remain
scattered across READMEs.

Naming, configuration conventions, and module boundaries show mild drift as a
result of past iterations. The storage model is inspectable but not yet
consolidated into a single reference. These gaps do not block use, but they
reduce the clarity, reproducibility, and ease of integration that FUTON1 is
intended to provide.
@next
Consolidate the documentation and contracts that define how text becomes graph state; stabilise the naming and configuration surfaces shared across CLI, API, and ingest workflows; and refine test coverage so the deterministic substrate remains reliable as modules evolve. Maintain FUTON1's minimal scope—storage, hydration, and inspection—while ensuring that higher futons can depend on its interfaces without ambiguity.

! instantiated-by: Prototype 0 — Baseline Deterministic Substrate [💤/已 ⛲/布]
+context: FUTON1 provides a deterministic storage substrate built from the demo/client workflow, the NLP interface, graph-memory, and XTDB persistence. These components together define the minimal end-to-end path by which text becomes durable graph state and can be inspected through the CLI or HTTP API.
+if: You want a stable understanding of what FUTON1 currently guarantees—its ingest pipeline, hydration behaviour, and contract surfaces—before extending or modifying any part of the stack.
+however: Documentation and examples are distributed across several READMEs; the overall "stack story" from ingest to XTDB is implicit rather than expressed as a single reference artifact, and the exact boundaries of the deterministic substrate are easy to lose track of during development.
+then: Capture a baseline snapshot of the current system: record a concise architecture note describing the pipeline (ingest → NLP → graph-memory → XTDB), freeze representative transcripts from the demo/client workflow, and

FUTON2

@multiarg futon2/devmap
@title FUTON2 Development Map
@audience futon-devs, simulation-agents, future-you
@tone formal-analytic
@style roadmap
@allow-new-claims true
@length any
@factor Energy (viriya)
**@IFR: FUTON2 is the viriya (energy) chamber: a narratable Active Inference engine whose ants are first demonstrators for AIF agent species that will eventually crawl FUTON1 graph memory, export state to higher futons, and turn insight into kinetic practice across the network.
@state Ant sim + AIF loop run end-to-end with analyzer hooks, but architecture docs, observation property tests, and hunger/tau goldens remain incomplete.
@next Capture the baseline architecture note, lock observation/property tests, and record hunger/precision goldens before layering new agent species.**

! instantiated-by: Prototype 0 — AIF Stack Baseline (observe → perceive → affect → core) [👕/卯 💤/已]
+context:FUTON2 runs a full Active Inference loop in the ant war-game, wiring 'ants.aif.observe', 'ants.aif.perceive', 'ants.aif.affect', and 'ants.aif.core' into one tick-by-tick pipeline.
+if:You want a stable story that shows how observation, prediction errors, precision, and actions integrate per tick.
+however:Modules are individually solid but no single document freezes hunger/tau semantics or traces the whole stack.
+then:Capture a top-level architecture note from sensory keys through world updates, lock current hunger and predictive micro-step parameters, and add four golden AIF microtraces.
+because:A documented baseline prevents accidental drift before you optimise policy, analyzers, or mechanics.

! instantiated-by: Prototype 1 — Observation Layer Hardening [🍲/则 👎/下]
+context: 'g-observe' normalises sensory evidence (food, pheromones, proximity, novelty, reserves) into the vectors passed downstream.
+if:You want deterministic observations across seeds and a nailed-down vector ABI for later ML.
+however:Normalisation bounds and neighbour logic assumptions are scattered across helpers.
+then:Write property tests that enforce 0–1 ranges, document the 'sense->vector' ordering, and add edge-case tests for grid borders, max-distance tweaks, and degenerate pheromone fields.
+because:Reliable observations keep perception and policy stable no matter the world configuration.

! instantiated-by: Prototype 2 — Predictive Coding Microcycle Clarity [💖/刊]
+context:Perception performs multi-step updates of mu, tau, Pi-o, goals, and weighted errors with annealing.
+if:You want inspectors (or future Arxana bridges) to understand belief updates.
+however:Hunger updates, sensory prediction, and error accumulation are interwoven without a canonical trace.
+then:Publish a single-tick perception trace with plots of hunger, tau, and error per micro-step, and formalise an interface for extracting a mu/precision snapshot.
+because:Clear cycles make later integrations with futon4 symbolic memory far easier.

! instantiated-by: Prototype 3 — Hunger & Precision Dynamics Goldens [🌲/高 💫/六]
+context:Functions such as 'tick-hunger', 'update-hunger', 'update-tau', and 'modulate-precisions' control the most sensitive dynamics.
+if:You want reproducible affective behaviour that survives refactors.
+however:There are no golden tests for tau evolution, hunger saturation, or reserve-driven adjustments.
+then:Record six golden sequences (normal, starvation, overstimulation, heavy-cargo, high-risk, high-ingest) and lock expected tau floors/caps for each.
+because:Once goldens exist, small tune-ups stop

FUTON3

@multiarg futon3/devmap
@title FUTON3 Development Map
@audience futon-devs, sandbox-agents, future-you
@tone formal-analytic
@style roadmap
@allow-new-claims true
@length any
@factor Joy / Rapture (pīti)
@IFR: FUTON3 is the place where the system turns messy activity into organised knowledge. It collects what happens, checks it against a library of shared patterns, and produces clear, trustworthy records that other futons can use to learn, improve, and act. FUTON3 makes everyday work auditable, meaningful, and easier to build on.

@state Transport runs reliably, events are logged, patterns can be checked, and Tatami sessions already feed structured evidence into the system. We have an early pattern catalog, working embeddings, and a first version of the workday pipeline. What remains is strengthening these links — tying checks into storage, producing clearer histories, and building the stable interfaces that other futons can depend on for coordinated reasoning.

@next Stand up the completed pattern canon + 'check!' API, standardise proof→graph/energy exports, and land the workday instrumentation + trail rollups as first-class MUSN services before expanding the flexiformal training ground.

! instantiated-by: Prototype 0 — Flexiformal Transport Baseline [💤/已]
+context:FUTON3 still rides the MUSN transport (WS+HTTP bus, SAFE/ADMIN REPL, NDJSON ingest), but its primary job is to host pattern checks and informal proof states, not free-roaming agents.
+if:You want an execution substrate where pattern obligations can be posed, evaluated, and recorded with deterministic transcripts.
+however:The baseline hello/event/session flows are documented as "sandbox trivia" instead of "proofwork IO contracts."
+then:Freeze the transport contract from the perspective of check jobs (pattern-apply, gap-report, trail-capture), provide golden transcripts for each, and version the 'sandbox' profile as the canonical proofwork runtime.
+because:Flexiformal checking needs a stable log format before any higher-layer reasoning can rely on it.
+evidence:docs/protocol.md, src/f2/router.clj, test/f2/router_test.clj.

! instantiated-by: Prototype 1 — Pattern Canon & Standard Library [💖/本 🙅/在]
+context:A small flexiarg/pattern library already exists (Nonstarter, Proto/Allow-Works, etc.) but is scattered between files.
+if:You want FUTON3 to be the place where patterns are curated with metadata, linking obligations, and proof status.
+however:There is no canonical store or schema for "pattern definitions + applicability notes + links to Futon devmaps."
+then:Stand up a 'f3.patterns' store (ID, clauses, references, pāramitā tags, fruits/orbs weights) populated with the current standard library and backfilled with source links.
+because:Without a canonical pattern store, the checker has nothing trustworthy to compare against.
+evidence: 'resources/pattern_store.edn', 'src/futon3/pattern_store.clj', and 'test/pattern_store_test.clj' keep the catalog synced with live tests + README references.

! instantiated-by: Prototype 2 — Similarity Field & Fake Embedding [💤/习]
+context:We already use a "fake embedding" (sigil adjacency) to find nearby patterns.
+if:You want reproducible neighbourhood queries for pattern suggestions.
+however:The embedding lives in ad-hoc CSVs and isn't wired into FUTON3's APIs.
+then:Integrate the sigil-distance matrix into FUTON3 as a first-class service (pattern search API, 'nearest-patterns' command, CLI/UI views) and publish tests proving deterministic neighbourhoods.

FUTON4

@multiarg futon4/devmap
@title FUTON4 Development Map
@audience futon-devs, pattern-agents, future-you
@tone formal-analytic
@style roadmap
@allow-new-claims true
@length any
@factor Tranquility (passaddhi)
**@IFR: FUTON4 becomes the passaddhi (tranquility) memory atelier where Arxana, graph memory, flexiarg, and agents interoperate, letting symbolic artefacts be edited, linked, and narrated in one continuous workspace that settles and integrates the energetic work beneath it.
@state Arxana reanimation through M4 is working (storage bridge, browsing, relations) while inclusion/transclusion UX, modern import/export shims, and QA/release prep remain open.
@next Deliver the pending inclusion UX, document modern import/export/migration shims, and grow the QA + contributor guide so a "revived" release can be tagged.**

! instantiated-by: Prototype 0 (Running Since 29 July 2025) [💗/从 🌏/甲]
+context:Prototype 0 establishes the symbolic foundations, artefact inventory, and DREAM logic.
+if:You want a coherent first cycle of a year-long 12-prototype run.
+however:Symbolic activation does not yet translate into memory operations or agent affordances.
+then:Establish baseline artefacts (README-manifesto, clock interface, Arxana reboot outline).
+because:This creates an anchor for all subsequent prototypes.

! instantiated-by: Prototype 1 — Arxana Reboot (Week-long target) [👎/新 🎋/长]
+context:Legacy literate-programming structure exists but is dormant.
+if:You want stable parsing, node navigation, and save-back-to-LaTeX working.
+however:Legacy Emacs-Lisp and refactored Clojure schemas diverge.
+then:Reboot spine parser, restore node browsing, define new nema-based schema, test round-trip edits.
+because:A functioning editor is the precondition for all downstream memory work.
+ next (workflow plumbing):
- Create import/export shims for modern doc formats.
- Produce migration notes for historical datasets.
+ next (QA & release prep):
- Grow the ERT suite (unit, integration, CLI smoke).
- Document test instructions and CI hooks.
- Provide contributor guide (coding style, tangling, testing).
- Tag a "revived" release with reproducible steps.

! instantiated-by: Prototype 2 — Graph Memory (Datascript) [💖/已 💤/生活]
+context:The hyperedge schema for Arxana exists on paper but not in running code.
+if:You want argument fragments to become addressable nodes with provenance.
+however:there is no minimal memory manager wiring Arxana buffers to Datascript.
+then:implement 'add-node', 'activate-node', and 'link-node'; persist them via Datascript; replay a session to prove round-trips.
+because:without a living graph memory, Arxana never escapes editor cosplay.

! instantiated-by: Prototype 3 — Flexiarg ⇄ Arxana Integration [💖/双]
+context:Flexiarg is acting as a pidgin for arguments and patterns.
+if:You want arguments to compile into graph nodes and be editable via Arxana.
+however:No translation layer exists between pidgin text and structured memory entries.
+then:Define schema mapping: argument-block → rule node (Q/A pair).
+because:This yields bidirectional editing: human-readable ↔ machine-structured.

! instantiated-by: Prototype 4 — Agent Layer Bridge (FUTON3) [📁/不能 👐/但]
+context:Agents can "speak" flexiarg but cannot yet access memory graphs.
+if:You want pattern-aware agents who can detect

FUTON5

@multiarg futon5/devmap
@title FUTON5 Development Map — Concentration, Nonstarter, Patterns of Improvisation
@audience futon-devs, improviser-agents, CT-metaweavers, future-you
@tone formal-analytic
@style roadmap
@allow-new-claims true
@length any
@factor Concentration (samādhi)
**@IFR: FUTON5 operates as the samādhi (concentration) engine, turning improvisation, CT riffs, musical chops, and Nonstarter economics into renewable practices that keep creativity gathered without teleological drag.
@state Design-only plan; prototypes have narrative intent but no running code or artefacts yet.
@next start**

! instantiated-by: Prototype 0 — Nonstarter Frame (Concentration without Teleology) [👴/乃 🍊/不能]
+context:FUTON5 is the samādhi layer: gathering energy around chosen patterns, sustaining focus, and refusing the teleology of success so value is created without capture and creativity concentrates without collapse.
+if:You want a layer that stabilises creative practice (music, coding, argument, CT diagrams) into repeatable, renewable flows.
+however:Current prototypes (MetaCA, flexiarg, musical studies, CT notes) sit in isolation and do not yet form a concentrated practice ecology.
+then:Define FUTON5 as the layer where prototypes from F4 are refined into systems—pattern libraries, CT riffs, musical grammars, improvisation logics, Nonstarter economics—so concentrated practice can run independent of outcome-chasing.
+because:Without a stable concentrator layer, FUTON4's tranquility and FUTON6's argument power cannot synergise.

! instantiated-by: Prototype 1 — Patterns of Improvisation (Music ⇄ Math ⇄ CT) [👈/归 🔃/世]
+context:Improvisation, category theory, MetaCA, and design patterns exhibit the same structural move: they generate new forms by transforming existing ones. FUTON5 requires not only descriptions of improvisation but *operators* that act on F4/F5 patterns.
+if:You want FUTON5 to unify musical, mathematical, and symbolic improvisation into a coherent substrate of transformation rules.
+however:Current improvisation notes, CT riffs, musical examples, and MetaCA sketches exist as fragments, not as typed operators that can act on pattern-graphs.
+then:Create a small "Patterns of Improvisation" protobook with 8–12 core operators (e.g., Prototype → Production, Kleisli Turn, Return to I, Phase-Switch, Noise Budget, Concentration Spiral). For each operator:
・ provide a riff (music),
・ a CT sketch (arrow diagram or endofunctor view),
・ a MetaCA local rule mapping,
・ and *crucially* define its action on F4/F5 pattern-graphs (input type, output type, invariants preserved, tensions altered).
Encode each operator as an F5 "meta-pattern node" consumable by F4 (memory) and F6 (argument).
+because:These meta-patterns form the initial FUTON5 "RNA pack": a finite alphabet of transformations that can reshape patterns, concentrate practice, and generate new improvisational structure across domains.

! instantiated-by: Prototype 1b — Sigil–Operator Lexicon (RNA Bootloader) [🈚/初 🌀/符]
+context:FUTON4 already uses a 256-sigil alphabet to index symbolic states and pattern traces. FUTON5 requires these sigils to index *meta-patterns*—operators that transform F4/F5 pattern-graphs.
+if:You want a finite, reusable, cross-domain alphabet of improvisational operators (an "RNA table") that can be invoked by agents, humans, and musical systems.
+however:Sigils currently tag patterns descriptively but lack operational semantics; no mapping exists between a sigil and the transformation it performs on a pattern-graph.
+then:Define a minimal operator spec:

FUTON6

@multiarg futon6/devmap
@title FUTON6 Development Map — Hyperreal Dictionary & Mathematical Habitat
@audience futon-devs, pattern-agents, HDM-stewards, future-you
@tone formal-analytic
@style roadmap
@allow-new-claims true
@length any
@factor Equanimity (upekkhā)
**@IFR: FUTON6 is the "mathematical habitat" layer of the stack: the place where
mathematics becomes inhabitable by humans
and agents through a pattern-indexed,
flexiformal, open-ended dictionary. It draws
verified argumentation from FUTON3;
structural memory from FUTON1; and provides
the conceptual terrain for FUTON7's
civic-scale reasoning. F6 is the **pattern-city of mathematics**, not the
execution engine (F3) nor the editing environment (F4).
@state Design-only vision; no dedicated HDM code or imports exist yet beyond conceptual outlines.
@next start**

! instantiated-by: Prototype 0 — FUTON6 Frame & Charter [💗/义 💖/今]
+context:F6 names the HDM layer: the place where mathematical content,
scholia, and argument patterns live as shared habitat.
+if:A crisp scope is needed so HDM does not dissolve into "general aspiration."
+however:Current F6 references still blur its role with F3 execution and F4 editing.
+then:Write the F6 charter: (1) remit, (2) neighbours (F3/F4/F5),
(3) responsibilities (stewardship of content, scholia, pattern indexing),
(4) non-responsibilities (checking = F3; tooling = F4).
+because:A charter stabilises FUTON's cascade and prevents scope creep.

! instantiated-by: Prototype 1 — Pattern/Proof Bridge [💖/入 💗/义]
+context:F3 handles flexiarg, proof trails, and pattern canonicalisation.
+if:F6 should receive arguments in a stable, machine-tractable form.
+however:F6 currently implies it must "perform" canonicalisation.
+then:Define a simple import contract: 'check → f6-node', which lifts checked
arguments from F3 into the HDM, attaching math metadata but not re-checking.
+because:This respects the separation of powers (F3 = execution; F6 = habitat).

! instantiated-by: Prototype 2 — Fruits/Orbs as Math-Annotators [💗/办 🍍/系统]
+context:F3 annotates flexiform arguments with fruits and pāramitā orbs.
+if:F6 wants to preserve narrative weight in math-native form.
+however:No mapping exists from fruit/orb → math-role (lemma, axiom, strategy).
+then:Define a light translation layer: e.g. 🍍→ indicator lemma; 🗝️→ axiom;
🍁→ strategic move; 🐉→ nontrivial dependence.
F6 stores these tags but
does not manage the fruit system itself.
+because:It preserves argumentative salience without overextending F6.

! instantiated-by: Prototype 3 — Hyperreal Dictionary Seedbed [💗/义]
+context:HDM's mission is to build inhabitable mathematics, not abstract criteria.
+if:F6 needs a concrete minimal slice to prove viability.
+however:No seed slice or exemplars have been frozen.
+then:Choose a coherent seed domain (e.g. compactness patterns in MSC26, basic
categorical arrows, Abelian groups, Dirichlet distributions) and create linked
entries: definitions, examples, small proofs, scholia, code.
+because:A tangible neighbourhood demonstrates HDM's inhabitable promise.

! instantiated-by: Prototype 4 — Layer-3 Reasoning Patterns [⛲/示]

FUTON7

@multiarg futon7/devmap
@title FUTON7 Development Map — Nomadic Civilisation & Civic Equanimity
@audience futon-devs, sphere-architects, future-you
@tone formal-analytic
@style roadmap
@allow-new-claims true
@length any
@factor Civic equanimity (upa-upekkhā)
**@IFR: FUTON7 holds the civic field of wisdom, translating futons 1–6 into sphere architectures, Amazon-breaker heuristics, and Hyperreal-derived seed institutions that cultivate equanimous, post-state civic life—the upa-upekkhā of institutions.
@state Design-only narratives outlining the civic field; no implemented prototypes or institutions yet beyond essays.
@next start**

! instantiated-by: Prototype 0 — Naming the Layer (Civic Field of Wisdom) [👴/田]
+context:FUTON7 designates the highest layer of the stack: the "martial application" of all prior futons toward the shaping of a post-state nomadic civilisation.
+if:You want an explicit locus for work that concerns culture, governance, civic habits, collective intelligence, non-state coordination, and world-building.
+however:F7 has been described indirectly (Amazon-breaker, Hyperreal as seed crystal, Sloterdijk's spheres) without a clear development remit.
+then:Define F7 as the civic field where equanimity (upa-upekkhā) is scaled from personal practice to institutional atmosphere; write its mandate: "Support distributed, non-state, resilient cultural formations with Hyperreal Enterprises as a seed instance."
+because:Naming the layer clarifies direction: F7 = how the world changes when Futons 0–6 are applied *socially* rather than individually.

! instantiated-by: Prototype 1 — Sphere Architecture (Sloterdijk Integration) [👴/无 🔃/门]
+context:Sloterdijk's spheres theory provides the conceptual grammar for collective enclosures, atmospheres, and shared immunological structures.
+if:You want F7 to model and cultivate "atmospheres of practice"—shared civic containers where reasoning, creativity, and care become normal.
+however:No translation exists between spherology and the FUTON stack; F7 has no "sphere-operators" yet.
+then:Define three sphere types for F7: 1) micro-spheres (dojo, band, mentoring dyads), 2) meso-spheres (Hyperreal teams, open research cohorts), 3) macro-spheres (post-state civic networks). Document how FUTON1–6 contribute to each sphere.
+because:Civilsation-scale practice begins with sphere-scaping—the deliberate shaping of atmospheres that support distributed wisdom.

! instantiated-by: Prototype 2 — Hyperreal Enterprises as Seed Crystal [👴/己 💗/义]
+context:Hyperreal Enterprises already functions as a micro-institution with values, narrative, economic flow, and client-based R&D loops.
+if:You want a living prototype that tests post-state organisational design in real economic terrain.
+however:Hyperreal's civic/institutional implications remain latent rather than formally articulated.
+then:Position Hyperreal explicitly as F7's seed crystal: a minimal self-governing unit practicing equanimous decision-making, distributed documentation, pattern-based governance, and nomadic economic strategy (consulting as caravan model).
+because:A concrete seed institution anchors F7's abstractions into lived practice and iterates the world-making in real time.

! instantiated-by: Prototype 3 — Amazon-Breaker Function [👴/引]
+context: "Amazon-breaker" describes technologies and social forms that prevent extractive centralisation, platform capture, and monoculture.
+if:You want F7 to actively resist the gravitational

+context: FUTON0 stabilises your epistemic rhythm: a weekly/multi-weekly cycle of review, planning, and salience-tracking that keeps work across all spheres coherent while avoiding data fragmentation or over-tracking.
+if: You want to programme yourself the way you programme systems—maintaining legible salients across institutional, consulting, technical, reflective, and infrastructural spheres with minimal friction.
+however: Without a normalised rhythm, weekly reviews slip; projects drift outside the canonical Org structure; and WIP silently grows past safe limits.
+then: Hold a weekly AOB/Ambitions sync; treat Org as the canonical plan/export store; mirror all active projects into a minimal FUTON1 substrate; review sphere balance weekly; and enforce the WIP cap (>3 active tracks requires pausing one). Cron/HUD integration surfaces upcoming reviews ("Weekly review in 3 days," "Monthly audit approaching") without additional lifelogging.
+because: A stable epistemic rhythm prevents unbounded cognitive load *and makes life feel lighter, more joyful, and more tractable*. When salients reappear on schedule and WIP remains bounded, drift does not accumulate and affect settles.
+evidence:
- Org already represents the canonical project hierarchy (AOB, Ambitions).
- Weekly AOB/Ambitions sessions are timestamped (Tatami/API or calendar).
- FUTON1 tags provide >90% work-session closure coverage.
- Sphere balance and WIP creep can already be inferred from Org + FUTON1.
+next: Consolidate the weekly sync into a fixed time; prune stale Org headings monthly; ensure every active project has an Org headline and FUTON1 tag; integrate HUD reminders for weekly/monthly cadence; and refine the WIP audit to produce actionable "pause/don't-start" prompts.
+next-evidence:
- Cron job generates weekly/mid-week/monthly reminders surfaced in the HUD.
- FUTON3 salience panel merges Org + FUTON1 metadata into a weekly
salients snapshot with diff from prior weeks.
- Quarterly rhythm analysis correlates stable cadence with subjective
affect measures (energy, focus, lightness) to test the experiential claim.

! instantiated-by: Prototype 2 — Core Obligations Contract [💤/业 💤/沒有]
@multiarg futon0/prototype2
@title Core Obligations Contract
@audience futon-devs, reflective-agents
@tone analytic-concise
@style flexiarg
+context: FUTON0 must uphold the fundamental constraints that keep the entire stack viable: sustained alignment with your current role, movement toward a more suitably aligned next role, and protection of basic health baselines. This layer also ensures that Prototype 1's analytics loop remains genuinely useful rather than self-perpetuating.
+if: You want your working life to remain healthy and coherent while moving toward a better-aligned future role, without sacrificing present obligations or drifting into avoidant behaviour.
+however: When attention shifts to higher layers, core obligations can slide into the background; health erosion becomes invisible; and epistemic-review routines can quietly expand beyond their real value.
+then: Maintain consistently and measurably strong alignment with the current role while increasing alignment with a future role; keep health baselines intact; and periodically check whether Prototype 1's analytics loop is delivering practical benefit.
+because: Upholding alignment and constraints at the base prevents the entire FUTON stack from drifting into incoherence or overload. When obligations, health, and review practices stay grounded, forward movement becomes stable and real.
+evidence:
- Health baselines visible through Prototype 0 (movement/sleep Y/N).
- Current-role performance inferred from deliverables, responsiveness,
and deadline stability.
- FUTON1 tags showing adequate work-session tag a reference point that reflects the present deterministic behaviour.
+because: A clear baseline makes the substrate's guarantees explicit, helps detect regressions in hydration or pipeline flow, and provides a stable reference against which future refinements or extensions can be evaluated.
+evidence:
- README documents deterministic demo/client behaviour, NLP interface,
graph-memory schema, and XTDB hydration.
- 'apps/demo', 'apps/client', 'apps/nlp-interface', 'apps/graph-memory', and 'apps/open-world-ingest' form the operational substrate.
- XTDB inspection and hydration behaviour are reproducible through existing
scripts and aliases.
+next: Compile the architecture note; select and store a minimal set of canonical transcripts; and publish them alongside a tagged release that marks the baseline behaviour.
+next-evidence:
- A dated, versioned snapshot containing architecture notes and
representative transcripts.
- CI or local reproducibility checks demonstrating identical XTDB and
focus-header outputs under the baseline tag.
- A single README-level reference pointing to the baseline for future
comparison.

! instantiated-by: Prototype 1 — Demo/Client Determinism & Focus-Header Instrumentation [💤/每 🚞/已]
+context: The 'apps/demo' and 'apps/client' workflow is the canonical entry point for interrogating the stack. It emits deterministic focus headers, hydrates from XTDB on startup, and exercises the NLP interface and graph-memory module together, exposing how ingest, mutation, and inspection behave in practice.
+if: You want the CLI and client workflow to act as reliable instrumentation for investigating hydration, salience updates, and focus-header formation— producing traces that can be compared across runs and environments.
+however: Existing smoke tests cover only narrow paths, and the behaviour of hydration, persistence, and focus-header emission under varied configurations is not yet exercised systematically. Legacy assumptions from earlier protocols still colour expectations about determinism.
+then: Establish a minimal determinism suite for the demo/client workflow: select a small set of representative sessions, fix their configuration, and record the expected XTDB and focus-header outputs. Treat these as canonical fixtures checked under CI so that changes in hydration, pipeline flow, or header structure become immediately visible.
+because: Treating the demo/client workflow as calibrated instrumentation—rather than an informal demo—provides a stable, reproducible surface for higher futons, lets developers detect drift in persistence or header logic, and clarifies exactly what guarantees FUTON1 provides.
+evidence:
- README documents deterministic demo/client behaviour, focus-header output,
and XTDB hydration.
- Golden fixtures already exist for graph-memory and NLP layers.
- XT inspection paths ('-M:storage/inspect') demonstrate reproducible
hydration and salience persistence.
+next: Curate a minimal determinism suite; freeze representative focus-header and XTDB outputs; and integrate these checks into CI as regression indicators.
+next-evidence:
- A small directory of canonical demo/client transcripts with corresponding
expected XTDB states.
- CI reports confirming stability of determinism across tagged commits.
- Clear README references linking these fixtures to FUTON1's guarantees.

! instantiated-by: Prototype 2 — NLP Interface Consolidation [🙍/申 📁/这些]
+context: 'apps/nlp-interface' provides deterministic NER/POS tagging and a small set of pattern-recognition hooks used across ingest and demo/client workflows. It is the shared entry point for turning raw text into structured tokens that the breaking entire colony behaviour.

! instantiated-by: Prototype 4 — Policy Layer Surfacing [🍲/十 😻/以]
+context: 'ants.aif.policy' selects actions using EFE metrics while 'attach-policy-diagnostics' emits traces.
+if: You want reproducible policy choices under controlled sensory inputs.
+however: There is no standalone harness that feeds synthetic mu/observation pairs.
+then: Build '(eval-policy mu prec obs config)' as a harness and add golden action rankings for ten canonical scenarios (e.g. rich food, enemy close, starvation risk).
+because: Stabilised policies are mandatory before benchmarking or bridging into futon1 salience services.

! instantiated-by: Prototype 5 — World Mechanics Contract (war.clj) [🌅/世界 🙂/但]
+context: The simulator handles evaporation, movement, gather/deposit, pheromone dynamics, hunger propagation, reserves, and termination.
+if: You want FUTON2 to be a reliable agent-environment backend.
+however: Movement heuristics, gather thresholds, and white-streak rules are powerful yet lightly documented.
+then: Document all invariants (pheromone caps, grid bounds, gather limits), seed movement decisions for determinism, and split movement modes into testable units.
+because: A clear contract keeps the world trustworthy when other prototypes depend on it for episodes.

! instantiated-by: Prototype 6 — Live Analyzer & Pivot Stream Stabilisation [🚢/在]
+context: The live analyzer consumes pivot events, computes EWMA metrics, detects starvation spirals, and prints mode transitions.
+if: You want analyzer outputs to feed futon3 evaluations or futon4 reflexive agents.
+however: Pivot schemas are implicit and analyzer assumptions are undocumented.
+then: Freeze a versioned pivot-row schema, add tests for required keys (:act, :mode, :cargo, :ing, etc.), and ship an example analyzer plugin for futon4.
+because: Pivot streams are FUTON2's event bus and must behave like a stable ABI.

! instantiated-by: Prototype 7 — Classic vs AIF Benchmark Suite [🗿/系統]
+context: Benchmark scripts compare classic ants to AIF runs across scores, durations, and G-ema.
+if: You want formal behavioural comparisons rather than anecdotes.
+however: Only three preset scenarios exist and there is no variance analysis.
+then: Add parametrised benchmarks spanning grid sizes, pheromone decay, hunger burn, and tau floors; capture plots and golden CSVs.
+because: FUTON2 should function as a scientific system with reproducible metrics.

! instantiated-by: Prototype 8 — External Control & Episode Export [⏏️/予 🤐/给]
+context: 'simulate' drives the war sim but offers no external stepping API.
+if: You want futon3 or futon4 to request "give me N steps with this config."
+however: Stepping remains tied to the CLI and HUD.
+then: Expose '(futon2.step world n {:emit-pivot? true})', add EDN exports of full episodes, and keep pivot batches consistent.
+because: Exported episodes are critical for agent training, memory, and pattern extraction.

! instantiated-by: Prototype 9 — futon1 ↔ futon2 Alignment Layer [💤/巳 📄/刊]
+context: FUTON1 emits focus headers and salience; FUTON2 emits percept/action streams.
+if: You want the ants to act as embodied readers + writers of futon1's graph.
+however: there is no shared dictionary between entities/intents and colony modes, so exports are useless to FUTON1.
+then: define the crosswalk (entities ↔ colonies, intents ↔ behaviours), publish an export schema, and prove it by replaying one colony while FUTON1 ingests the stream and surfaces it back to users.
+because: a working bridge proves ants can reason over the same facts as the rest of the stack.

+because: Pattern suggestion quality underpins the flexiformal workflow and must be inspectable.

! instantiated-by: Prototype 3 — Applicability Engine & Check DSL [👐/么 💖/什么]
+context: We want FUTON3 to answer "can this pattern apply?" and "what new obligations arise?"
+if: You want a declarative DSL (possibly REPL-safe) for posing check contexts and receiving structured verdicts (applies / blocked / missing evidence).
+however: Checks today are conversational heuristics in aob-chatgpt or handwritten flexiargs.
+then: Define a 'check!' API (inputs: pattern ID, context EDN, evidence refs; outputs: status, missing fields, derived tasks), enforce schema validation, and log every check as a proof state record.
+because: Turning patterns into executable checks is what makes FUTON3 "flexiformal" rather than just narrative.
+next: Ship the Futon1/Futon2 adapters so 'futon3/logs/checks.edn' entries replicate into graph + viriya stores instead of living only in local logs, and expose '/musn/check' over HTTP once the transport contract is frozen.
+evidence: 'src/futon3/checks.clj', 'src/f2/router.clj', and 'test/transport_test.clj' (check route cases) exercise the DSL end-to-end.

! instantiated-by: Prototype 4 — Trail & Proof-State Journal [💖/八 💖/弓]
+context: Wisdom trails already exist for tracing exploration, but now they must document proof obligations and how work sessions discharged them.
+if: You want every check, dismissal, and new pattern idea to leave an auditable trail.
+however: Trails are still general-purpose exploration logs, not proof-state graphs.
+then: Extend the trail schema with ':pattern/id', ':obligation/id', ':action/tags', and ':delta/joy', add rollups that show which devmap clauses advanced per day, and expose a '/musn/trails/proof' export for futon4 archives.
+because: The whole point is to measure how daily practice advances the devmaps.
+ design: The Tatami HUD cue-reflection pattern mirrors every FROM-CHATGPT entry back through '/musn/cues' so fruits/orbs in the HUD match the transport's cue embedding ('contrib/aob-chatgpt.el:1503', 'src/f2/ui.clj:97', 'src/futon3/cue_embedding.clj:214').
+evidence: 'resources/tatami-events.edn' captures live tatami sessions, 'resources/tatami-context.edn' logs every FROM-CHATGPT-EDN HUD payload, and the cue embedding tests ('test/futon3/cue_embedding_test.clj') prove keyword/intentionally-null payloads round-trip.
+ blocked-by: [:transport-persistence :cue-validation]
+next: (1) persist cue annotations into the trail export so Prototype 4 records which fruits/orbs were actually computed, and (2) add proof artifacts (HUD screenshots or '/musn/cues' transcripts) that demonstrate the cues signal real changes instead of static defaults before declaring the prototype stable.

! instantiated-by: Prototype 4b — Intent Embedding & Salience Map [💡/八 💡/弓]
+context: Prototype 4 turns trails into proof-state journals, but it still treats fruits/orbs as cosmetic tags instead of Kolmogorov-style proof types for each constructive step.
+if: You want the HUD to say not just "something happened here" but "this was an indicator check, and it was (or wasn't) aligned with your values" — i.e. to classify 'intent → pattern reasoning' as typed proof steps like 🍌→🔵 rather than anonymous log lines.
+however: The current trail schema records ':pattern/id' and ':obligation/id' but has no machinery to interpret 'intent → pattern reasoning' as a constructive proof step, embed it in fruit/orb space, and ask "what kind of operator was this?" (fruit) and "what meta-property did it exhibit?" (orb) — e.g. "🍌: indicator construction, 🔵: virtue-aligned".
+then: Define a futon3 embedding pipeline that maps each ':trail/intent' plus ':pattern/reasoning' tuple into a low-dimensional vector, projects that vector onto a (fruit, orb) pair understood as (operator-class, meta-property), and computes a ':rule/salience' score from structural, pragmatic, and reflective components; write ':fruit/id', ':orb/id', and ':rule/salience' back onto the trail events so Prototype 4 can roll up "which

tension, link motifs, and apply PIDGIN→PROSE.
+however: No API exists to let a GPT-based agent query the graph.
+then: Create REPL-level bridge (datascript queries + JSON-rpc + Emacs functions).
+because: This is what upgrades GPT into a pattern-capable agent.

! instantiated-by: Prototype 5 — Multi-user Memory (MUSN revival) [🔦/在 🏡/但]
+context: MUSN existed as a concept for multi-user semantic memory.
+if: You want collaborators (Serena, Rob, Harvinder, etc.) to share a semantic space.
+however: No multi-user or sync layer exists yet.
+then: Use Datascript local + XTDB remote as stepping stones.
+because: This will open the door to multi-agent co-reasoning and shared prototyping.

! instantiated-by: Prototype 6 — Argumentarium / Scholium Mode [🌲/木 🌳/但]
+context: You want something like Spinoza's *Ethics* but with imports, hypergraphs, and links.
+if: You want a library of arguments with provenance, alternative proofs, counterproofs.
+however: Current flexiargs do not yet link into a live scholium tree.
+then: Add scholium metadata + "view scholium tree" mode in Arxana.
+because: This becomes the living mathematical-philosophical substrate.
+ next (inclusion/transclusion UX):
- Visually differentiate included vs. transcluded regions in the main article buffer.
- Provide toggle commands to highlight each derivative type independently.
- Document the visual affordances and add a regression test covering both styles.

! instantiated-by: Prototype 7 — XTDB Integration [⌛/久 💖/长]
+context: Bi-temporal, multi-user storage needed for long-term evolution.
+if: You want durability, branching histories, provenance queries.
+however: XTDB idioms differ from Datascript; schema migration needed.
+then: Rekey schema, add transaction logs, implement time-travel queries.
+because: This turns Arxana into a serious memory engine.

! instantiated-by: Prototype 8 — Pattern Library Activation [💤/但 💖/却]
+context: You have dozens of patterns (Mojo, ORP, Paramita, Flexiarg, etc.).
+if: You want agents to recommend patterns or detect pattern-relevant situations.
+however: Patterns are not yet encoded in memory or linked to triggers.
+then: Import patterns as graph rules; map IF/HOWEVER/THEN/BECAUSE into triggers & actions.
+because: This yields genuine pattern-aware intelligence.

! instantiated-by: Prototype 9 — StackExchange Import [💖/又 🙎/无]
+context: StackExchange holds thousands of argued Q/A pairs that fit Arxana's schema.
+if: You want to harvest them as production rules with provenance.
+however: there is no importer aligning their Markdown into our graph schema.
+then: build a parser, map Q/A + comments into rule nodes, annotate tags, and cross-index them with existing flexiargs.
+because: seeding the memory with curated Q/A pairs gives Arxana instant argumentative density.

! instantiated-by: Prototype 10 — Corpus Mathematics Layer [👯/但 🐺/却]
+context: You want to bridge natural/formal logic (Strand 1B).
+if: You want Dirichlet distributions and Abelian groups to live in the same memory model.
+however: No symbolic math integration exists yet.
+then: Build math namespace + flexiarg-lemma type + REPL test harness.
+because: This is how FUTON becomes an "AI Scientist."

! instantiated-by: Prototype 11 — Literary Interface (Penmaster) [💖/好]
+context: You want Arxana to also serve as a book-editing and generative-literary tool.

• input type (pattern, riff, argument, or multi-node diagram),
• output type (new pattern-graph, mutated pattern, or practice directive),
• graph effect (context shift, force modulation, solution re-expression, example migration),
• musical / CT / practice phenotype,
• collapse modes (misuse, teleology traps).
Bind 8–12 sigils to the operators defined in Prototype 1 ("Patterns of Improvisation") to create the first FUTON5 RNA lexicon.
Store the mapping in Arxana as editable F5 nodes so that new operators can be added, tuned, or retired.
+because: Without a sigil-indexed transformation alphabet, FUTON5 lacks executable concentration mechanics. With it, sigils become active operators—an RNA engine capable of transforming DNA-patterns from FUTON4 into living, improvisable forms.

! instantiated-by: Prototype 2 — Concentration Engine (Chops, Not Goals) [🙍/工 💖/已]
+context: Concentration in F5 is about capacity, not outcomes: sustained work on craft (music, CT, code) without dangling the carrot of "success."
+if: You want a system for renewable, non-self-punishing discipline.
+however: Current practice cycles depend heavily on mood, projects, or institutional demands.
+then: Establish a "Concentration Engine" protocol with 3–5 stable practice grooves (bass technique, CT reading, coding drills, MetaCA tinkering), ensure each groove has low activation energy and clear termination points, and track them in FUTON5 as patterns rather than habits.
+because: Concentration emerges from rhythm, not obligation.

! instantiated-by: Prototype 3 — Pattern Library (Post-Flexiarg) [💖/但是 💬/世]
+context: Flexiarg formalises arguments (F6) while FUTON5 formalises patterns.
+if: You want a domain-agnostic, cross-modal pattern language scaffold that sits between Arxana and the argumentarium.
+however: Existing patterns are scattered across papers (P4P, P4D, P4NG) and are not harmonised.
+then: Define a minimal F5 pattern schema (@pattern-id, IF/HOWEVER/THEN/BECAUSE where useful, examples across music/code/CT/life, invariants/failure modes, "improvisation affordances") and refactor 6–12 core patterns into that schema.
+because: FUTON5 needs a pattern substrate that can plug directly into FUTON6's argument logic.

! instantiated-by: Prototype 4 — MetaCA as Improvisation Physics [🙍/己 🌀/我]
+context: MetaCA is your 2014 prototype of local-rule self-evolution and mirrors your 2025 simulation-based self-understanding.
+if: You want to reinterpret MetaCA as the "physics" of improvisation: everything evolves locally but is constrained globally by a deep rule (Dhamma / AIF).
+however: MetaCA code exists, yet its conceptual integration with FUTON5 is not explicit.
+then: Reframe MetaCA as an F5 conceptual engine where local rules equal riffs/CT arrows/pattern transforms, the global rule embodies AIF, edge-of-chaos represents productive creativity, collapse modes name burnout/ego/overcontrol, and capture this in one short essay inside the F5 devmap.
+because: This gives FUTON5 a mathematical "feel" that is embodied, generative, and non-institutional.

! instantiated-by: Prototype 5 — Nonstarter Social Physics [💰/无 🙍/元]
+context: Nonstarter = anti-teleology, refusal of capture, joy in giving, Patti Smith energy, LANDBACK aesthetics.
+if: You want FUTON5 not only to concentrate craft but to concentrate value without reproducing institutional extraction.
+however: There is no current model linking creative concentration with non-extractive flows.
+then: Define a simple Nonstarter Ledger inside F5 that tracks "value given" (hours, riffs, patterns released), "value received" (collaboration, uplift, resonance), enforces a "no-ask posture" protocol, encodes reciprocity loops based on pattern contribution rather than money, and distinguishes RESCUE requests from COLLABORATION requests.

+context: HDM is the choreography of mathematical moves, not just a pile of entries.
+if: FUTON6 needs a vocabulary of cross-domain reasoning idioms (quotient tricks, local->global moves, dualisation, extremisation) so agents can navigate strategy.
+however: No curated Layer-3 catalogue exists, so everyone re-invents its own outline language.
+then: Freeze a small but expandable L3 pattern set, tag the seed domain entries with those moves, and expose L3 IDs to F6-aware agents as their default reasoning vocabulary.
+because: L3 patterns supply the "physics" of mathematical reasoning and make HDM truly traversable.

! instantiated-by: Prototype 5 — Layer-4 Domain Patterns [🌳/本]
+context: Subject areas (MSC slices) have canonical machinery—ultrafilters, Yoneda, compact operators, Markov kernels—that sit between general motifs and specific entries.
+if: FUTON6 needs Layer-4 pattern nodes that capture domain-specific constructs, typical lemmas, and dependencies.
+however: Today those structures are either flattened into individual entries or treated as vague generalities.
+then: Identify 10-20 L4 candidates for the seed slice, represent each as a node with domain metadata, canonical usages, and referenced L3 moves, and ensure L4 nodes refactor like any other content.
+because: L4 patterns provide the local physics that make each mathematical neighbourhood habitable.

! instantiated-by: Prototype 6 — L3<->L4 Relation Map [〰️/互]
+context: FUTON6 spans HDM motifs, L3 reasoning physics, L4 domain machinery, and concrete entries.
+if: Imports and viewers need an explicit handshake so layers do not blur.
+however: No rulebook specifies how L3 motifs embed in L4 constructs or how L4 clusters point back to universal moves.
+then: Publish the relation: L3 = field-independent moves, L4 = field-dependent machinery; require new entries to declare which layer they instantiate, and build viewers showing the L3<->L4 wiring for any definition, proof, or example.
+because: A fractal, Alexandrian cascade keeps FUTON6 coherent as it scales.

! instantiated-by: Prototype 7 — HDM Dojo Mode [👩‍👧/友]
+context: Mathematics becomes native when practiced like a dojo: progressive disclosure, sparring partners, and navigable landscapes.
+if: FUTON6 is to turn math into a habitable habitat for humans and agents.
+however: Current assets are static entries; they do not yet function as a training ground that links moves, contexts, views, and guidance.
+then: Blend L3/L4 tagging with scholium landscapes, progressive outline->detail modes, and agent sparring protocols so users can practice reasoning moves inside HDM itself.
+because: Mathematical fluency emerges from embodied practice, not passive reading; the dojo keeps HDM alive.

! instantiated-by: Prototype 8 — Scholium & Proof-Landscape Mode [💗/示 🙎/义]
+context: F3 emits proof trails; F1 provides graph memory.
+if: F6 wants users/agents to traverse mathematics as connected terrain.
+however: No math-specific landscape viewer exists.
+then: Add scholium metadata (e.g. related lemmas, commentary nodes, viewpoint shifts) and render "neighbourhoods": clusters of results, arguments, patterns, counterexamples.
+because: Landscapes differentiate HDM from a static encyclopedia.

! instantiated-by: Prototype 9 — Language ⇄

pull toward state-like apparatuses and exploitative superstructures.
+however: No operational definition of "Amazon-breaking" exists within the FUTON stack; it risks becoming a slogan rather than a design constraint.
+then: Define Amazon-breaker heuristics (e.g. decentralised memory, pattern-encoded governance, nomadic modularity, multi-agent redundancy, civic equanimity, reverse-platformisation). Tag any F6/F5/F3 work that strengthens these heuristics.
+because: Anti-monopoly structural feedback must be designed in from the start; otherwise F7 collapses back into the state form.

! instantiated-by: Prototype 4 — Civic Equanimity Protocol (Upa-upekkhā at Scale) [💖/弓]
+context: You have identified F7 with the Pāramitā of Equanimity: balance-beyond-balance, the stability of clarity under pressure.
+if: You want this Pāramitā to inform collective, not just individual, behaviour.
+however: No mechanism yet translates individual reflective rhythm (Futon0, PARamita cycles) into group-level civic practice.
+then: Design a Civic Equanimity Protocol: shared PAR cycles, distributed pattern reviews, collective decision check-ins (fruits/orbs used as civic signals), and non-reactive governance rituals adapted from tai chi application.
+because: Institutions that act with even-minded care are the defining signature of a future civilisation that is neither authoritarian nor chaotic.

! instantiated-by: Prototype 5 — Multi-Sphere Learning Ecosystems [🙍/女 💏/孑]
+context: You have real data from ORP workshops, PLACARD, band rehearsals, strategic consulting, dojo practice, and peer mentoring.
+if: You want F7 to articulate what a mature, distributed learning ecosystem looks like.
+however: These practices are scattered across domains and not formalised into a unified civic-learning grammar.
+then: Identify recurring multi-sphere motifs (shared inquiry, pattern elicitation, improvisational coherence, mutual regulation); map them to F5 patterns and F6 reasoning operations; prototype a "Nomadic Learning Cell" drawing from Hyperreal + ORP + dojo + band.
+because: Nomadic civilisations endure by transmitting culture horizontally, not vertically; multi-sphere learning is their engine.

! instantiated-by: Prototype 6 — Economic Nomadism & Post-State Viability [🐺/但]
+context: Future nomadic groups will require economic resilience untethered from state sponsorship, salary structures, and bureaucratic scaffolds.
+if: You want F7 to model a viable economic substrate that is anti-fragile, creative, and regenerative.
+however: Current income streams (job, consulting, side projects) are not yet framed as a prototype nomadic economy.
+then: Define a minimum viable nomadic economy: (1) consulting caravans (Hyperreal), (2) intellectual capital loops (F6 argumentarium + patterns), (3) low-cost, high-leverage creative production (band, book, prototypes), (4) geographic and institutional mobility.
+because: F7 must be economically real, not merely philosophically interesting.

! instantiated-by: Prototype 7 — Interoperability with State Forms without Capture [👶/长]
+context: Post-state formations will necessarily interoperate with existing state institutions (universities, funders, cultural organisations).
+if: You want to retain freedom while collaborating fruitfully.
+however: Nomadic groups risk being absorbed, bureaucratised, or instrumentalised by state forms if guardrails aren't explicit.
+then: Establish an F7 "Porous Interface Charter" clarifying how Hyperreal and related projects interact with institutions: export patterns, consulting, co-reasoning, but preserve governance, culture, and memory internally.
+because: Being permeable without becoming captured is essential for long-term autonomy.

! instantiated-by: Prototype 8 — Civilisational Simulations (Nomadic Scenario Work) [🙍/扎]

closure coverage.
+next: Make alignment in the current role more explicit (quick weekly pulse-check); establish a small, low-friction trajectory toward the next aligned role; and add a brief weekly check on the usefulness of Prototype 1.
+next-evidence:
- Cron-backed extraction of role-search traces (calendar items, files,
emails) to show gentle forward movement.
- A FUTON3 prompt asking “Did this week’s review help maintain alignment?”
as a simple measure of analytics-loop value.
- Correlations between Prototype 0 vitality signals and stable current-role
performance to validate the alignment hypothesis.

! instantiated-by: Prototype 3 — Ambition 0 Meta-Managed Infrastructure [🙇/叉 🌅/每]

@multiarg futon0/prototype3
@title Ambition 0 — Meta-Managed Infrastructure
@audience futon-devs, reflective-agents
@tone analytic-concise
@style flexiarg
+context: FUTON0 needs an infrastructure layer that not only keeps reality and representation aligned, but also increases the system’s capacity for alignment over time (Ashby-style requisite variety).
+if: You want your tools to help you notice, raise, and refine signals that support increasing alignment in work, health, and direction.
+however: Interfaces drift, signals blur, and representational structures lose fidelity. When this happens, the system cannot increase alignment because it cannot perceive what is changing or what needs attention.
+then: Treat FUTON1 as the source of truth; use Org as a lightweight interface to it; prune dashboards to highlight only rising or fading salients; and run a brief reconciliation loop that raises proto-pattern signals for review.
+because: This layer provides the Markov blanket through which alignment can improve. Prototype 3 acts as the experience-design prototype of FUTON3: it raises signals, surfaces proto-patterns, and increases representational variety, while FUTON3 will later formalise this as proof-theoretic salience certification.
+evidence:
- Org and FUTON1 currently reflect overlapping state that allows detection
of divergence.
- Dashboards already reveal what is becoming noisy or inert.
- Backups and timestamps show which elements are live or stale.
+next: Migrate remaining project state into FUTON1; reduce Org to a thin interface; and introduce a weekly “rising/fading signals” pane in FUTON3 to surface proto-patterns for reflection.
+next-evidence:
- Divergence checks that quantify where representation is lagging reality.
- Signal-rise/signal-fade detectors for dashboards and headings.
- FUTON3-generated proto-pattern stubs based on behavioural traces.

! instantiated-by: Prototype 4 — Ambition 1 Hyperreal Enterprises Practice [👴/工作 👎/两]
@multiarg futon0/prototype4
@title Ambition 1 — Hyperreal Enterprises Practice
@audience futon-devs, reflective-agents
@tone analytic-concise
@style flexiarg
+context: Hyperreal requires a consulting practice that increases alignment between your research, your livelihood, and your long-term direction while generating reusable structure rather than bespoke one-offs.
+if: You want each engagement to contribute both to financial stability and to the evolution of a sustainable, research-informed independent practice.
+however: Without intentional structure, communication slips; deliverables fragment into bespoke artefacts; and the insights generated fail to loop back into the system that needs them.
+then: Keep client communication transparent; produce deliverables as reusable, documented modules; and extract consulting insights as proto-patterns that flow into the Hyperreal backlog and FUTON3’s salience pipeline.
+because: This turns consulting into an alignment engine: work generates reusable rest of FUTON1 can rely on without ambiguity.
+if: You want a single, well-defined NLP module whose behaviour is testable, documented, and independent of the surrounding CLI or transport logic.
+however: The current README gives only a minimal overview of the pipeline, and the boundaries between tokenisation, tagging, chunking, and downstream pattern recognition remain implicit in code rather than expressed as a clear API. Tests cover specific fixtures but not the full staged behaviour.
+then: Make the pipeline stages explicit: document and expose the tokenise → tag → chunk → recognise flow, write unit tests for each stage, and provide a brief, focused README describing the module’s public API, expected inputs and outputs, and how upstream components should depend on it.
+because: A consolidated and documented NLP interface provides a stable grounding for ingest and demo/client workflows, reduces accidental coupling, and makes it easier to extend or revise individual stages without destabilising the rest of FUTON1.
+evidence:
- README describes deterministic NER/POS behaviour and shared pipeline usage.
- Test runners exist for NLP fixtures (‘apps/nlp-interface/test’).
- Pattern-recognition hooks are already used in ingest and CLI flows.
+next: Clarify the staged pipeline, expand tests to cover each stage, and produce a concise README that reflects the actual API and expected data shapes.
+next-evidence:
- Stage-by-stage unit tests passing under CI.
- Updated README documenting pipeline flow and public functions.
- Agreement across ingest and demo/client modules on API usage.

! instantiated-by: Prototype 3 — Graph-Memory API & Seed Graph [💤/已 👐/么]
+context: ‘apps/graph-memory’ defines the Datascript schema, salience metadata, and store management functions that underpin FUTON1’s fact graph. It coordinates in-memory state with XTDB persistence and is the central place where entity and relation documents are shaped before being written or read.
+if: You want FUTON1 to expose a clear, documented contract for how entities, relations, and salience fields are represented and mirrored between Datascript and XTDB, so that other futons and external tools can rely on the same facts without depending on internal implementation details.
+however: The existing README and tests describe schema and storage behaviour but do not yet present a concise API-level view of graph operations. Helper functions are still oriented toward REPL use, naming conventions vary, and there is no single description of what is considered part of the persistent “seed graph” or which documents are guaranteed to be mirrored into XTDB.
+then: Promote ‘graph-memory’ to a documented API: define the core entity and relation operations (init, upsert, link, query), state explicitly which fields and document types are mirrored into XTDB, and describe any default seed graph that is expected to exist in a fresh store. Align naming and function boundaries with this contract and treat incompatible changes as migrations rather than ad-hoc edits.
+because: A clear graph-memory contract turns FUTON1’s internal schema and store logic into a reliable interface. This allows higher futons and external tools to depend on the same graph representation, reduces ambiguity about what is persisted, and makes changes to the underlying storage behaviour visible and deliberate.
+evidence:
- README documents the Datascript schema, XTDB touchpoints, and hydration
behaviour used by FUTON1.
- Test suites for graph-memory exercise persistence and salience fields.
- XT inspection paths demonstrate how entity and relation documents are
stored and reloaded.
+next: Extract and name the core graph-memory operations as a public API, document which document types and fields are mirrored into XTDB, and describe any expected seed graph in the module README.

! instantiated-by: Prototype 10 — Curriculum Scenarios & AIF Behaviour Cards [🎴/乡 🙅/归]
+context:The current war scenario is rich but single-purpose.
+if:You want FUTON2 to double as a pedagogical engine for Active Inference.
+however:There is no named curriculum.
+then:Create behaviour cards (home-return, starvation spiral, over-exploration, defensive lattice, greedy gatherer collapse), simulate each with fixed seeds, and export the episodes.
+because:These cards seed pattern extraction and future training systems.

! instantiated-by: Prototype 11 — AIF Agent Debugger & Mu/Precision Visualiser [😻/跟 🐺/一]
+context:Pivot logs are textual and require manual parsing.
+if:You want interactive introspection of mu vectors, tau evolution, and weighted errors.
+however:No visualiser or “follow Alice” mode exists.
+then:Build a Clerk notebook (or similar) that plots tau, hunger, errors, and pivot timelines, plus a mode that traces one agent’s full mu timeline.
+because:Visual insight accelerates iteration and debugging.

! instantiated-by: Prototype 12 — Year-End Packaging & Story [💑/末 🌀/年]
+context:FUTON2 must read as a coherent simulation layer to partners and future prototypes.
+if:You want it to be adoptable as a research simulator.
+however:The codebase is powerful but intimidating without a unifying story.
+then:Package the release with a top-level README, whitepaper, stepping API, goldens, and versioned pivot schema.
+because:Strong packaging turns FUTON2 into a reusable, inspectable module inside the full FUTON stack.

! instantiated-by: Prototype 13 — Graph Memory Adapter & Agent ABI [💤/已 👎/甲]
+context:FUTON1 now offers a documented graph-memory API; FUTON2 needs to speak it to graduate beyond the sandbox.
+if:You want ants to be the first drop-in “AIF agent” that reads fact graphs and writes back its beliefs.
+however:percepts/actions remain locked to grid coordinates, so nothing maps to FUTON1 entity IDs or relation semantics.
+then:design a bidirectional adapter (percepts → graph facts, graph cues → priors), version the ABI, and replay a colony run that exports beliefs into FUTON1 salience tables.
+because:once the adapter exists, future AIF agents inherit a proven contract instead of bespoke glue.

! instantiated-by: Prototype 14 — Viriya Field & Social Activation Metrics [👕/卯 ⚡/介]
+context:FUTON2 carries the Factor of Awakening “energy/viriya”: vigor, commitment, and kinetic practice.
+if:You want simulations to reflect and inform real social activation—networked collaboration, shared commitments, and practice loops across humans + agents.
+however:Current metrics stop at hunger/tau; nothing measures vigor, pledge strength, or coordination throughput.
+then:Introduce viriya indicators (e.g., colony activation score, pledge adherence, transfer of trails into collaborative tasks), log them per episode, and export them so futon3/futon7 can read “energy health.”
+because:Tying the simulation to viriya turns FUTON2 into a practice accelerator instead of a curiosity.

! instantiated-by: Prototype 15 — Sandbox Bridge & Presence Interface [🌀/内 👕/里]
+context:FUTON3 now emphasises flexiformal proofwork but still hosts multi-user rooms and SAFE REPL affordances.
+if:You want AIF agents (ants and successors) to inhabit those rooms, receive room/talk updates, and emit actions back into the simulation.
+however:No interface maps MUSN room events/presence into FUTON2 percepts, nor do ants announce themselves inside FUTON3.
+then:Define a bidirectional “presence bus” (rooms → observation cues, agent actions → MUSN events), add authentication so only devmap clauses advanced, via which kinds of proof step (e.g. 🍌 indicator definition or check, 🍒 simple inference, 🍊 breakdown into a proof plan, etc.), and how strongly they mattered” per day.
+because:In Kolmogorov’s sense, a proof is a construction of an object; here, each trail step is a constructive proof step that builds a pattern-typed object whose canonical form is its (fruit, orb) type. Typing steps like “🍌→🔵” lets you distinguish “this is just a check” from “this is an indicator that actually preserves the virtue invariant,” and salience separates merely valid rules from those that are central to your meta-theory in practice.
+ design: Implement ‘futon3.intent-embedding/intent->vec’ to encode ‘:trail/intent’ + ‘:pattern/reasoning’ as a vector, ‘futon3.intent-embedding/vec->fruit+orb’ to pick the nearest (fruit, orb) type under a basis where fruits encode proof operator classes (e.g. 🍌 = indicator/check, 🍐 = invariant-preserving step) and orbs encode meta-properties (e.g. 🔵 = value-aligned / virtue-preserving, 🟠 = boundary-discernment); define ‘futon3.intent-embedding/vec->salience’ to combine geometric proximity to an “ideal” direction for each (fruit, orb) type with usage frequency and an optional reflective score. Extend the trail export with ‘:fruit/id’, ‘:orb/id’, ‘:rule/salience’, and add HUD affordances so Tatami can sort, filter, or highlight proof-state entries by their typed role (e.g. all 🍌→🔵 value-aligned checks) instead of just by timestamp.
+evidence:Early Cue Embedding tests (‘test/futon3/cue_embedding_test.clj’) already show that keyword payloads round-trip through the embedding; hand-labelled fruit/orb assignments in ‘resources/paramita-grid.edn’ provide a gold set for checking whether ‘vec->fruit+orb’ recovers the expected operator/meta-property types on known patterns, including basic Komolgorov relations like “indicator-construction steps that should land in the 🔵 virtue band”.
+next:(1) Generalise the existing cue embedding to operate on ‘:trail/intent’ + ‘:pattern/reasoning’ tuples and return (fruit, orb) as proof step types, (2) run a calibration pass over a small set of manually typed patterns to tune the fruit/orb basis and salience weights, ensuring that canonical relations like 🍌→🔵 (“value-aligned indicator checks”) land in the intended region, and (3) wire the enriched trail export into Prototype 4’s rollups so devmap progress can be inspected as “which constructive proof-types actually fired today, how salient they were, and whether the checks that claimed to align with your values really did so in the embedding.”

! instantiated-by: Prototype 5 — Workday Instrumentation & aob-chatgpt Bridge [💖/弓 👯/几]
+context:Within ‘aob-chatgpt’ we already experiment with reflecting on daily actions against devmap obligations.
+if:You want FUTON3 to serve as the backend for those reflections.
+however:There is no official API for submitting “workday claims,” mapping them to patterns, or receiving verification feedback; roles of “workday participants” and “instrumentation stewards” are undefined.
+then:Publish a ‘workday/submit’ endpoint (inputs: timestamp, activity, evidence links), run it through the check DSL + embedding, respond with pattern hits/misses plus follow-up obligations, and name the actors explicitly (participants supply evidence; instrumentation stewards curate dashboards); integrate the same flow into the ChatGPT tooling.
+because:This is how empowered action (“joy”) becomes measurable proof progress and how role responsibilities stay legible.
+next:Replace the temporary ‘futon3/logs/workday.edn’ queue with a Futon1 persistence call (‘workday->graph’) and add HTTP affordances + dashboards so instrumentation stewards can audit submissions without tailing logs.
+evidence:Workday intake lives in ‘src/futon3/workday.clj’ with coverage in ‘test/transport_test.clj’ (workday submission) and README sections describing the API.

! instantiated-by: Prototype 6 — Pattern Creation Workbench [📁/当 🚴/手]
+context:New patterns emerge from trails and devmap edits, but the capture is currently manual

+if:You want to edit Penmaster or draft your book via Arxana blocks.
+however:No good UI for long-form text yet.
+then:Add “manuscript mode” + chunking + cross-reference map.
+because:FUTON becomes the writing environment.

! instantiated-by: Prototype 12 — Year-End Synthesis [💗/末 🌽/年]
+context:The prototype cycle ends; reflection and forward-planning needed.
+if:You want long-term stability, and upgraded beta-schema.
+however:Accumulated divergence between prototypes must be unified.
+then:Run full PAR + CLA + DREAM integration cycle.
+because:This sets the stage for FUTON5+.

+because:With a functional Nonstarter economy, FUTON5 becomes spiritually and materially sustainable.

! instantiated-by: Prototype 6 — Improvisation Metrics (Not KPIs) [👈/归 😻/叉]
+context:Metrics destroy creativity when tied to teleology.
+if:You want a way to sense concentration without collapsing into institutional logic.
+however:No metrics currently track musical, cognitive, or conceptual “chops.”
+then:Define a set of “Improvisation Indicators” (Noise Budget, Flow Duration, Surprise-to-Stability Ratio, Return-to-I Coherence, Cross-Domain Lift between music/writing/coding) and track them qualitatively in F5 without gamification.
+because:FUTON5 needs feedback, not surveillance.

! instantiated-by: Prototype 7 — F4⇄F5⇄F6 Conduction Pathways [👐/什么]
+context:F4 is tranquility, F6 is war machine, and F5 is the samādhi bridge.
+if:You want to flow smoothly between relaxed inscription (F4), concentrated craft (F5), and structured reasoning (F6).
+however:There is no explicit pathway linking the three.
+then:Define three conduction modes—F4→F5 (prototype → practice, what to work on today), F5→F6 (practice → argument, what becomes a formal insight), F6→F5 (argument → pattern, what needs to be embodied)—and document these flows as short “transmission patterns.”
+because:Without conduction, the stack fragments instead of concentrating.

! instantiated-by: Prototype 8 — 12-Prototype Year (Concentration Arc) [👴/年 💑/岁]
+context:You are already running a 12-prototype year.
+if:You want FUTON5 to shape it into a concentration arc rather than a sequence of builds.
+however:There is no formal mapping of prototypes to F5 themes.
+then:Assign each prototype a concentration domain (MetaCA, Patterns of Improvisation, CT chops, Musical chops, Flexiarg → pattern transforms, Nonstarter ledger, Pattern economy, F5 system interfaces, Meta-Improvisation, AIF integration, Relational improvisation patterns, Year-end coherence synthesis) and keep the map current as the year unfolds.
+because:Concentration deepens through cyclical return.

! instantiated-by: Prototype 9 — Yearly F5 Review (Concentration Renewal) [👴/口 🌝/这样]
+context:Concentration weakens if not refreshed.
+if:You want F5 to stay light, concentrated, and improvisational rather than dogmatic.
+however:Without review, patterns accrete into heaviness.
+then:Run a yearly cycle that prunes dead patterns, refines musical grammars, updates CT riffs, consolidates improvisation patterns, refreshes the Nonstarter ethos, and removes anything that smells like KPIs.
+because:Concentration equals aliveness, and aliveness must be renewed.

Math Transformations [💬/文 💬/更]
+context:F6 aspires to maintain both NL-friendly and math-native views.
+if:You want consistent transformations across entries.
+however:NL ↔ math translations are currently ad hoc.
+then:Define several translation templates for the seed domain (e.g.,
“definition block”, “example pattern”, “lemma skeleton”) and exercise them
with FUTON3’s flexigrammar.
+because:Even partial bilingualism greatly improves usability.

! instantiated-by: Prototype 10 — Math-Aware Agent Protocol [💗/义 🙅/分]
+context:F6 will eventually host agents that annotate, refactor, or query entries.
+if:You want safe and productive agent behaviour.
+however:The current protocol is conflated with F3’s checking API.
+then:Define an F6-specific protocol: allowed queries (definitions, pattern IDs,
dependency graphs), acceptable outputs (annotations, reorganisations),
and guardrails (no overwriting canonical entries without review).
+because:Math-literate agents are central to HDM-as-habitat.

! instantiated-by: Prototype 11 — Non-Capture Guardrails [🐊/刊]
+context:HDM inherits the “war machine” mandate: remain non-state, non-enclosed.
+if:F6 is to remain an open mathematical habitat.
+however:Capture risks (institutional, corporate, epistemic) are unarticulated.
+then:Publish a Non-Capture Charter specifying: open licensing, provenance
transparency, anti-proprietary entanglement, dual-representation redundancy.
Mark entries whose sources impose constraints.
+because:Commons integrity is crucial for HDM’s long-term survival.

! instantiated-by: Prototype 12 — F6⇄F7 Scenarios [💗/化 🐜/支]
+context:FUTON7 builds civic imaginaries and long-range scenarios.
+if:Math should inform F7, not remain siloed.
+however:There is no pipeline from HDM entries to F7 narratives.
+then:Package small argument bundles (e.g. growth/decay patterns,
equilibrium models, scaling laws) as exportable F6→F7 kits.
+because:Mathematics should feed collective imagination, not remain inert.

! instantiated-by: Prototype 13 — FLEX Training Ground Link [💗/新]
+context:F3 now ships the “Flexiformal Training Ground.”
+if:F6 wants stewards who know how to handle patterns and proofs.
+however:F6 does not yet reference this training layer.
+then:In onboarding docs, require that new math agents/humans pass the F3
training ground. Capture their trails and store them as scholium notes
attached to early entries.
+because:This closes the developmental loop and ensures quality control.

+context:F7 needs a way to test future structures before implementing them.
+if:You want to prototype nomadic civic forms as thought-experiments and fictional/technical hybrids.
+however:No structured simulation practice exists across scenarios.
+then:Develop 2–3 scenario bundles (e.g. “Nomadic Math Sangha,” “Distributed Research College,” “Hyperreal Field Office in the Zone”) and tie each to relevant F6 argument bundles and F5 pattern clusters.
+because:Scenario practice turns abstract philosophy into actionable civic design.

! instantiated-by: Prototype 9 — Year-End Nomadic Governance Audit [👴/其 💗/要]
+context:Cultures need periodic tuning to maintain coherence and avoid drift.
+if:You want F7 to stay aligned with equanimity, abundance, independence, and anti-capture heuristics.
+however:No cycle currently exists that evaluates Hyperreal, the prototypes, and the emerging civic sphere as a coherent F7 entity.
+then:Create an annual F7 governance audit, integrating: (1) PARamita maps, (2) fruit/orb data, (3) economic logs, (4) Hyperreal practice notes, (5) scenario updates, (6) stack-wide tensions.
+because:A post-state civilisation must cultivate its own continuous reflective governance ritual—its civic meditation.

structure, structure generates higher variety, and higher variety increases your future alignment opportunities. Hyperreal becomes evolutionary rather than transactional.
+evidence:
- Existing consulting deliverables already include modular components.
- Client communication typically happens through channels that are easily
timestamped and reviewable.
- Notes from past engagements show extractable conceptual structure.
- The VSATLAS work demonstrates a clean example: a reusable flexiarg design
that extends the client's system while revealing future service-provision
revenue streams.
+next: Introduce a lightweight "module template" for new deliverables; add a brief reflection step at the end of each engagement; and feed extracted insights into FUTON3 for proto-pattern surfacing.
+next-evidence:
- Detection of repeated module reuse across engagements.
- FUTON3-generated proto-patterns based on recurring consulting moves.
- Evidence of alignment gain: smoother scoping, clearer pricing,
better-fit clients, and reduced bespoke overhead.

! instantiated-by: Prototype 5 — Ambition 2 FUTON–Arxana Stack Stewardship [💖/用 🚴/书]
+context: FUTON0 needs a reliable, steadily improving environment for thought. Daily use and clear architectural notes keep the FUTON–Arxana stack coherent and usable.
+if: You want FUTON–Arxana to become a dependable workspace for reasoning, development, and collaboration.
+however: Skipped daily use weakens the system; undocumented decisions cause drift; and premature collaboration introduces scope creep.
+then: Use FUTON1 daily as the live testbed; record architecture reflections as they arise; and add collaborators gradually with guardrails that protect scope and focus.
+because: Frequent contact and lightweight documentation prevent drift and make the system stronger over time. This keeps FUTON–Arxana evolving into a stable, coherent environment rather than an abandoned prototype.
+evidence:
- FUTON1 already supports daily usage with minimal friction.
- Architecture notes demonstrate where documentation has prevented drift.
- Early collaborative experiments (e.g., with Rob) show the value of guardrails.
+next: Stabilise daily usage with light scaffolding; formalise a minimal architecture note-taking protocol; and extend Arxana gradually as stable components emerge.
+next-evidence:
- Daily FUTON1 usage detectable via timestamps.
- Architecture notes showing reduced drift or clearer decision points.
- Arxana prototypes yielding stable nodes or literate fragments.

! instantiated-by: Prototype 6 — Ambition 3 Mojo, Evolving Variety, + Reflective Rhythm [💖/弓 🌅/当]
+context: FUTON0 needs a source of evolving, meaningful variety. Moments of "wondrous apprehension"—small encounters with insight, novelty, or strangeness—keep the internal model flexible and alive.
+if: You want learning, creativity, and self-understanding to deepen through controlled contact with the unknown rather than through routine or ritual.
+however: When these encounters vanish or are postponed indefinitely, patterns ossify, curiosity shrinks, and alignment stops increasing.
+then: Invite small, natural opportunities for wondrous apprehension—through ideas, conversations, creative acts, or symbolic cues—and metabolise them lightly, without turning them into obligations or scheduled rituals.
+because: These brief encounters act as evolutionary impulses: they expand perspective, refresh the generative model, and increase adaptive variety. This introduces just enough uncertainty to stimulate growth without destabilising the system.

+next-evidence:
- An updated 'apps/graph-memory' README that presents the API, mirroring
rules, and seed-graph expectations in one place.
- Tests that target the documented API functions rather than only internal
helpers.
- A short migration note for any changes that affect stored documents or
XTDB layout.

! instantiated-by: Prototype 4 — XTDB Operations & REPL Workflows [🐜/分 🐺/甲]
+context: FUTON1 mirrors all graph mutations into XTDB and provides a stable hydration path so that entity and relation documents, salience fields, and type data persist across restarts. The repository already includes REPL notes, inspection snippets, and a dedicated '-M:storage/inspect' entry point for examining the store directly.
+if: You want developers to inspect, query, and verify the persistent store using a small set of predictable entry points, without relying on ad-hoc XTDB queries or bespoke REPL sessions.
+however: Common inspection tasks—listing entities, examining relations, checking salience fields, or understanding schema layout—still require manual XTDB queries or scanning several snippets across READMEs and test files. These workflows are reliable but not yet consolidated into a minimal operational interface.
+then: Package the existing XTDB inspection workflows into a small set of reusable functions (e.g., list-entities, list-relations, inspect-entity, counts), present them in one place with brief examples, and highlight the salience-survives-restart demonstration as a canonical illustration of the hydration contract. Treat these recipes as the reference interface for inspecting and understanding FUTON1's persistent state.
+because: Consolidating XTDB operations into clear, reusable workflows reduces friction, makes store inspection more consistent, and strengthens FUTON1's role as a deterministic storage substrate rather than an opaque backend. Developers gain a predictable way to understand how mutations appear in XTDB and how hydration restores them.
+evidence:
- 'README-storage.md' documents XTDB inspection behaviour and usage.
- The '-M:storage/inspect' alias provides automated examples of querying the
store.
- Hydration and salience persistence are demonstrated through existing
scripts and XT touchpoints.
+next: Surface the same XTDB schema and inspection workflows in Emacs so FUTON3 can browse persistent store contents without relying on filesystem paths or ad-hoc REPL interaction.
+next-evidence:
- A simple Emacs view that displays XT entities, relations, and salience
metadata using the documented inspection functions.
- Consistent behaviour between the Emacs view and the REPL inspection entry
points.

! instantiated-by: Prototype 5 — Open-World Ingest Workflows [🔦/世界 🍲/开]
+context: 'apps/open-world-ingest' streams arbitrary text through a deterministic CoreNLP pipeline and writes entities, mentions, and relations into XTDB. These documents become part of the same persistent store that the demo/client workflow hydrates on startup, allowing background knowledge to accumulate outside interactive sessions.
+if: You want FUTON1 to support both offline corpus ingestion and interactive exploration, with both surfaces reading from and writing to the same XTDB state in a predictable way.
+however: Ingest currently operates as a standalone CLI with limited guidance on how its outputs integrate with the demo/client view of the world. Directory conventions, expected data-root layouts, and test coverage for ingest semantics (ID stability, relation structure, negation handling) are not yet consolidated.
+then: Define a standard data-root layout for ingest and demo/client workflows, document how ingest-produced XTDB documents appear in the hydrated store, and provide small example workflows that ingest a text sample and then query the resulting state through the demo/client

designated colonies bridge, and ship a demo where ants report discoveries into a FUTON3 trail.
+because:This proves that AIF agents can show up wherever proofwork happens, not just inside a war grid.

! instantiated-by: Prototype 16 — Lobby Bus & MUSN Intake [🌀/内 📬/门]
+context:FUTON3 runs with MUSN as a shared graph and chatgpt-shell as a HUD, but ChatGPT-API currently has only conversational access and no principled way to commit events back into MUSN.
+if:You want ChatGPT-API to propose or make *persistent* updates (breach logs, paramitā orbs, devmap notes, pattern instantiations) without giving it full god-mode over the stack.
+however:There is no explicit one-room "Lobby" where HUD traffic is normalised into MUSN events, no single-writer policy, and no clear schema for what counts as an escalated, storable event versus transient chatter.
+then:Create a dedicated Lobby room in MUSN with a minimal intake schema (':musn/events' envelopes on an escalated bus), grant exactly one writer identity (the chatgpt-shell / FUTON3 service account), and implement a Clojure bridge that (a) parses HUD EDN, (b) validates and timestamps Lobby events, and (c) transacts them into the MUSN graph as append-only nodes that downstream agents can subscribe to.
+because:This proves that a single HUD-bound agent can safely "step into" MUSN through a constrained door—acting as a semi-privileged clerk rather than an all-powerful daemon—and sets the pattern for later extending the presence bus (Prototype 15) to multiple rooms and multiple authenticated agent writers.
+ example: To assist the user with time management, ChatGPT-API offered: "Do you want breaches auto-logged if you're still active at 22:10?" At present there's no way for ChatGPT to *actually* log such breaches anywhere. Rather than assuming all escalated events must be elicited in the same way at all times, allow 'set-user-system-prompt' in futon3's 'aob-chatgpt.el' to switch between modes (e.g. ':normal', ':breach-logging') under external control (cron, or a MUSN agent). ChatGPT-API can then *propose* mode shifts or breach events via the Lobby, while the MUSN side decides which proposals to enact.
+ example2: While reviewing a day's evidence, ChatGPT-API proposes a structured "paramitā reflection" event via the Lobby:
```
{:musn/events
[{:event/type     :paramita/reflection
:event/paramita :equanimity
:event/notes    " User can apply 9 of Cups to understand the source of a workplace frustration."
:event/request  :musn.store!}]}
```
This safely introduces reflective practice artefacts into MUSN without conflating them with actual mode changes or breach logging. It demonstrates that the Lobby is not only
for operational concerns (like curfew rules) but also for epistemic/ritual
artefacts that support the broader FUTON practice space. The HUD then renders a subtle, clickable banner: Reflection exercise suggested ▷

! instantiated-by: Prototype 16 — Wisdom Trail Harvest & Pattern Seeding [💖/去 👶/所]
+context:FUTON3 trails now record which patterns were checked or advanced during human work sessions.
+if:You want FUTON2 analyzers to ingest those trails, compare them to colony behaviour, and commission new agent goals.
+however:There is no ingestion path for trail EDN, nor a feedback loop that says "agents, go gather evidence for Pattern X."
+then:Add a 'trail->mission' translator that turns FUTON3 proof deltas into FUTON2 goal queues, plus a reporting job that shows how agent runs affected subsequent checks.
+because:Shared missions keep the "energy" loop tight between simulations and human proofwork.

in futon4.
+if:You want FUTON3 to draft candidate patterns/flexiargs from accumulated proof states.
+however:Manual backlog in FUTON4 creates tension; there is no readiness window for when drafts are "hand-off ready."
+then:Build a summariser that clusters trails, extracts hooks, and emits draft flexiarg blocks (IF/HOWEVER/THEN/BECAUSE) tagged with suggested pāramitā, ready for futon4 editors, and declare readiness criteria (e.g., at least N supporting checks + steward review) so handoffs are predictable.
+because:FUTON3 should not just consume the library; it should propose expansions grounded in lived work while signalling when drafts can be trusted.

! instantiated-by: Prototype 7 — Joy-as-Empowered-Action Metrics [💖/弓 🚴/力]
+context:Joy here = Spinozan increase in power to act, evidenced by commitments you can now keep.
+if:You want metrics that show whether checks are empowering or draining.
+however:Current "joy metrics" counted UI friction; they did not measure proofs discharged or obligations unblocked.
+then:Track 'time-to-proof', 'obligations-cleared', 'follow-on commitments', and 'viriya linkages' per session, and visualise them alongside subjective notes.
+because:Empowered action is the KPI for this futon.

! instantiated-by: Prototype 8 — futon2/futon1 Proof Hooks [👕/当 💤/已]
+context:FUTON3 already generates proof states; FUTON1 (graph) and FUTON2 (energy) need those proofs as fuel.
+if:You want every accepted check to become a graph relation and a viriya delta without manual glue.
+however:today the exports are bespoke Clerk notebooks or one-off scripts.
+then:standardise two adapters— 'proof->graph' (materialise relations with provenance) and 'proof->energy' (emit activation deltas for futon2 dashboards)—and lock them with fixtures + schemas.
+because:proofs only matter when other futons can act on them automatically.
+next:Wire the Tatami event log into FUTON1 (per 'futon3/tatami_store.clj' TODO) so session events flow into graph-memory before we expose the adapters.

! instantiated-by: Prototype 9 — Proof Training Ground ("Flexiformal Agent Sandbox") [📁/义 💗/门]
+context:FUTON3 needs a reproducible training ground where humans and agents can practice flexiformal checks before contributing to FUTON6.
+if:You want collaborators (including FUTON6 math stewards) to experience the proof workflow hands-on.
+however:README/docs still emphasise ants and MUD metaphors and do not ship a simulator/tutorial bundle; roles (training facilitators vs. FUTON6 learners) and release readiness are not declared.
+then:Package a "Flexiformal Training Ground" release: updated README, tutorials referencing 't4r/' training patterns, sample check sessions, and a minimal simulator that feeds FUTON6's Hyperreal entries; name facilitators (FUTON3 maintainers) and consumers (FUTON6 trainees), and publish readiness criteria (tutorials complete, simulator recorded, onboarding docs updated) before tagging the release.
+because:Storytelling plus an explicit training ground turns FUTON3 into a reproducible service that upstream layers (FUTON6, FUTON7) can consume and steward.
+next:Aggregate Tatami session proof events into 'proof-chain.edn' (per 'tatami_store.clj' TODO) so the training ground exports persistent transcripts.

+evidence:
- Rhythm experiments such as PAR sessions, RAP sessions, Mojo prompts, and
card draws have reliably triggered moments of insight without becoming rigid.
- Creative or relational encounters have historically produced generative
perspective shifts.
- Minimal reflection has been sufficient to integrate these signals.
+next: Keep the system open to small shocks of insight; capture only what feels naturally significant; and allow FUTON1 or FUTON3 to mark these moments lightly when they matter.
+next-evidence:
- Occasional tags noting spontaneous insight or perspective shift.
- Detectable reductions in stagnation or pattern ossification.
- FUTON3 identification of proto-patterns emerging from these encounters.

! instantiated-by: Prototype 7 — Ambition 4 Role Transition Trajectory [🔦/口 💑/资料]
@multiarg futon0/prototype7
@title Ambition 4 — Applied Futon Theory: Co-Evolution With Future Environments
@audience futon-devs, reflective-agents
@tone analytic-concise
@style flexiarg
+context: FUTON0 must support your co-evolution with future environments—roles, collaborators, and communities that do not yet fully exist but already exert directional influence as futons (real-but-virtual futures).
+if: You want your future working life to emerge through mutual shaping rather than reactive job-seeking or inherited scripts.
+however: When futures are treated as fixed targets or when symbols are mistaken for the futons they mediate, misalignment grows and trajectories become brittle.
+then: Attend to symbolic signals that reveal possible future environments; explore lightly; cultivate relationships that show reciprocal responsiveness; and let alignment emerge through ongoing co-evolution rather than forced planning.
+because: Futons influence the present something like photons. The co-evolution of present and future echoes Gotama's dependent causality—agent and future conditions arise together in a logical chain—and Alan Moore's insight that all times are immediately accessible in certain states of consciousness. In applied futon theory, alignment grows through co-evolution with virtual states, not through selection or prediction.
+evidence:
- Prior trajectories (Hyperreal, VSATLAS, ORP, band work) emerged through
reciprocal adaptation, not explicit search.
- Symbolic cues—patterns, conversations, prototypes—have repeatedly revealed
access to futures not yet actual.
- Misaligned contexts typically show low responsiveness early, acting as
non-resonant futons.
+next: Track signals of futonic resonance or dissonance; maintain light exploratory contact with promising environments; and let the trajectory refine itself through reciprocal influence rather than pressure.
+next-evidence:
- FUTON1 tags capturing futonic moments (resonance, responsiveness,
momentum).
- Notes marking when symbolic cues open or close access to future
attractors.
- Increasing clarity, lightness, and alignment as co-evolution proceeds.

! instantiated-by: Prototype 8 — Stack Coverage Check [👐/广 💤/卜]
+context: Devmaps belong to FUTON3: they are symbolic membranes through which FUTON3 perceives futons and maintains structural coherence across the stack. FUTON0's role is to support the mindful conditions that allow these symbolic structures to appear, evolve, and be refreshed.
+if: You want FUTON3 to notice where the system's symbolic interfaces to future environments are missing, stale, or drifting, before the whole system loses alignment.
+however: When narrative, reflection, and or API surfaces. Add a minimal set of golden tests covering entity-ID stability and relation formation.
+because: When ingest and interactive exploration share a clear, documented storage path, FUTON1 can accumulate durable background knowledge in a controlled way. This makes the persistent world state coherent across offline and online workflows and maintains determinism across restarts.
+evidence:
- README confirms deterministic ingest via CoreNLP and XTDB mirroring.
- Demo/client workflows hydrate from XTDB on startup and therefore can read
ingest-produced documents.
- Existing golden fixtures demonstrate deterministic behaviour in other
pipeline components.
+next: Document shared data-root conventions; provide an ingest→query example in the README; and implement golden tests for ingest ID stability and relation structure.
+next-evidence:
- A worked example showing ingest followed by demo/client queries against
the same XTDB store.
- Golden tests verifying stable entity IDs and relation outputs for a fixed
input.
- README updates linking ingest workflows to the persistent world model.

! instantiated-by: Prototype 6 — Thematic Clusters & Intent Metadata [🙅/无 💤/上]
+context: FUTON1 can attach lightweight thematic tags to utterances or text spans during ingest or interactive use. These tags—derived from deterministic NLP features or simple pattern-recognition hooks—provide a minimal layer of "intent metadata" that downstream futons may use for analysis or policy.
+if: You want FUTON1 to expose a small, deterministic set of thematic categories that describe common conversational moves or content types, without relying on probabilistic classifiers or embedding higher-level inference in this layer.
+however: There is no consolidated thematic taxonomy, the tags used in logs are not documented as part of the public contract, and no examples show how these tags appear in the stored XTDB documents or hydrated in-memory state. Integration points within demo/client or ingest workflows remain implicit.
+then: Define a minimal set of thematic clusters supported by deterministic NLP features, document them as part of FUTON1's public contract, and ensure that produced tags (when enabled) appear predictably in both in-memory and XTDB representations. Provide a small set of example workflows illustrating how these tags are attached during ingest or interaction and how they can be queried or inspected.
+because: Lightweight thematic metadata strengthens FUTON1's role as an inspectable storage substrate. By keeping the taxonomy small and deterministic, FUTON1 avoids embedding inference or policy while still providing structured signals that higher futons may use for reasoning or pattern-based behaviour.
+evidence:
- README confirms deterministic NLP and pattern-recognition hooks.
- Logs produced by demo/client workflows already contain descriptive
metadata for some utterances.
- XTDB persistence and hydration ensure that any added metadata can be made
durable and inspectable.
+next: Specify a small, deterministic thematic taxonomy; document how tags appear in FUTON1 outputs; and add examples showing ingestion or interaction annotated with these clusters.
+next-evidence:
- Updated README describing thematic clusters and their data shapes.
- Deterministic tests verifying tag presence and structure.
- Example ingest→query or demo/client transcripts showing how tags persist.

! instantiated-by: Prototype 7 — CLI Ergonomics & Multi-Instance Isolation [🎏/资料 🍒/示]
+context: The demo/client workflow exposes bang commands, flags, and environment variables for selecting data directories, overriding XTDB configuration, and controlling persistence behaviour. These surfaces already allow multiple

wondrous apprehension are weakened, futons become harder to perceive; devmaps stagnate; and FUTON3 loses access to future-relevant structure.
+then: Maintain a light narrative awareness of the system’s evolving futures; allow futons to be apprehended through conversation, reflection, and symbolic cues; and let FUTON3 formalise these perceptions into updated devmaps when needed.
+because: FUTON0 provides the experiential ground—a mindful openness to futons— while FUTON3 handles representational coherence. Without the former, FUTON3 cannot see; without the latter, FUTON0’s perceptions cannot stabilise. Together, they keep the stack aligned across time.
+evidence:
- Narrative reflection in this conversation has repeatedly revealed futons
that were not yet encoded in devmaps.
- FUTON3 updates gain clarity when experiential insight precedes them.
- Wondrous apprehension consistently precedes formalisation.
+next: Support FUTON3 by staying attuned to emerging futons; mark futon moments lightly when they arise; and trust FUTON3 to handle structural consolidation.
+next-evidence:
- FUTON1 entries capturing futon moments.
- FUTON3 updates that align with experiential insight.
- Clearer, more resonant devmaps appearing naturally over time.

instances of FUTON1 to run side-by-side on the same machine.
+if: You want developers and tools to run isolated demo/client or API sessions for testing, experimentation, or production-like scenarios without interfering with each other’s storage or configuration.
+however: Best practices for using ‘BASIC_CHAT_DATA_DIR’, XTDB resource overrides, ‘config.edn’, and reset/compact/export options are spread across READMEs and command-line examples. It is easy to misconfigure data roots or XTDB paths, and no single reference describes the recommended conventions for running multiple instances safely.
+then: Document a small set of configuration patterns (e.g., dev/test/prod data roots), clarify how env vars and config.edn interact, and provide brief examples of running multiple isolated instances. Add smoke tests that launch separate demo/client or API sessions with distinct data roots. Include a simple ‘/status’ helper or equivalent CLI flag that reports the active configuration.
+because: Clear ergonomics and isolation practices make FUTON1 safer to embed in tooling and CI workflows, reduce accidental data mixing, and strengthen the determinism and reproducibility guarantees laid out in FUTON1’s IFR.
+evidence:
- README documents environment overrides, config.edn usage, and the
structure of XTDB data directories.
- Demo/client and API workflows can already be run with separate data roots.
- XTDB instances behave independently when configured with distinct paths.
+next: Provide documented configuration patterns, add isolation-oriented smoke tests, and implement a ‘/status’ helper that surfaces the active config.
+next-evidence:
- A README section showing dev/test/prod data-root conventions.
- Automated smoke tests confirming independent operation of concurrent
instances.
- A ‘/status’ output (CLI or API) showing current data-root and XTDB config.

! instantiated-by: Prototype 8 — Golden Scripts & Determinism Checks [🐜/不要 🌱/久]
+context: FUTON1 already uses scripted EDN conversations and golden outputs to verify deterministic behaviour in the demo/client workflow, the NLP interface, and graph-memory operations. These fixtures ensure that focus-header formation, XTDB hydration, and pipeline flow remain stable across revisions.
+if: You want confidence that changes to FUTON1 do not introduce regressions in determinism, header structure, or end-to-end behaviour, and that core workflows continue to perform within reasonable runtime bounds.
+however: Golden coverage remains incomplete. Existing fixtures focus on specific modules rather than end-to-end flows, and there is no consolidated suite capturing short conversational runs, longer histories, or ingest-driven scenarios. Runtime behaviour is observable but not yet checked in a structured way.
+then: Curate a small suite of canonical golden scripts that exercise representative workflows: a short demo/client session, a longer session with history, and a minimal ingest→query example. Record their expected outputs and basic runtime characteristics, and run them under CI so deviations in XTDB state, focus-header structure, or runtime ranges are immediately visible.
+because: A compact, well-chosen golden suite makes FUTON1’s determinism guarantees explicit, stabilises expectations about performance, and turns the substrate into a measurable layer rather than an informal demonstration.
+evidence:
- README documents deterministic behaviour, hydration patterns, and golden
fixtures already present in module tests.
- NLP and graph-memory test runners use scripted inputs and expected outputs.
- XTDB inspection tooling (‘-M:storage/inspect’) demonstrates reproducible
state across runs.
+next: Assemble the canonical golden suite, add runtime checks under CI, and link the suite from the top-level README as FUTON1’s determinism

reference.
+next-evidence:
- A small directory containing canonical scripts and their expected outputs.
- CI jobs reporting on determinism and basic runtime stability.
- README references explaining how to run and interpret the golden suite.

! instantiated-by: Prototype 9 — External Client & API Contract Clarification [📁/无 🙅/已]
+context: FUTON1 already exposes an HTTP API (‘apps/api’) that lets external tools query and update the persistent store through the same command-service logic used by the demo/client workflow. This provides a minimal surface for submitting utterances, inspecting state, and retrieving deterministic summaries such as focus headers.
+if: You want external clients to interact with FUTON1 without scraping CLI output or depending on internal modules, and to rely on a stable, documented interface for obtaining focus-header data and querying the underlying graph.
+however: The available endpoints and their expected inputs and outputs are documented across several READMEs rather than in a single reference, and there are no minimal example clients showing how to issue requests and interpret the deterministic responses. The boundary between FUTON1’s API and higher-level agent behaviour remains implicit.
+then: Consolidate FUTON1’s HTTP API contract: document the endpoints, describe the structure of request and response bodies (including focus headers), and provide one or two minimal example clients that demonstrate how to submit an utterance and inspect the resulting state. Keep the API minimal and deterministic, leaving policy or agent behaviour to higher futons.
+because: A clear, documented API surface allows external tools to rely on FUTON1’s deterministic storage and summarisation behaviour without embedding orchestration logic. This strengthens FUTON1’s role as a stable substrate while avoiding scope creep into agent design or protocol negotiation.
+evidence:
- README documents the existence of the HTTP API and describes how to start
it and configure its datastore.
- Focus headers and entity summaries are already emitted by the same
command-service used by the API.
- XTDB-backed state is stable and consistent across CLI and API surfaces.
+next: Produce a concise API contract, document the shape of focus-header responses, and include minimal example clients demonstrating deterministic interaction.
+next-evidence:
- A single API reference file enumerating endpoints and data shapes.
- Sample clients (Clojure and one other language) communicating with the API.
- CI tests confirming that responses remain stable across versions.
- **Clients that omit profile configuration correctly resolve to the
documented default profile, and the ‘/status’ surface shows the resolved
datastore path.**

! instantiated-by: Prototype 10 — World-State Access & Storage Model Documentation [💖/已 💤/本]
+context: FUTON1 maintains the canonical world state for the stack through its combined Datascript and XTDB representation. Downstream tools—whether FUTON4, Arxana, Emacs interfaces, or ad-hoc scripts—need a clear understanding of how this state is structured and how to retrieve entities, relations, salience fields, and other stored documents without relying on internal assumptions.
+if: You want any external consumer to be able to inspect FUTON1’s world state deterministically, using documented queries and data shapes rather than reverse-engineering internal code paths or storage conventions.
+however: Although the README describes persistence and hydration behaviour, there is no consolidated description of the storage model: which document types exist, how entities and relations are shaped in XTDB, how salience metadata is encoded, or which example queries reliably retrieve these structures. Export options

exist, but no single reference explains their intended use.
+then: Publish a concise storage-model reference that defines FUTON1’s persisted document types, key fields, and their relationships. Provide example XTDB and Datascript queries for retrieving entities, relations, salience metadata, and type information. Illustrate how downstream tools can consume this world state without depending on internal module boundaries.
+because: A clear storage-model contract makes FUTON1 a dependable foundation for the rest of the FUTON stack. It allows external tools to read and reason over the same durable world state without special-case integrations or export modes, preserving determinism and reducing the risk of accidental divergence.
+evidence:
- README documents XTDB hydration, document structure, and the relationship
between XT persistence and Datascript caching.
- XT inspection workflows (‘-M:storage/inspect’) already expose entity and
relation documents.
- Demo/client workflows and ingest pipelines both write into the same store.
+next: Produce a dedicated storage-model reference; add example XTDB and Datascript queries; and clarify how exported snapshots or raw XTDB directories may be consumed by external tools.
+next-evidence:
- A single document outlining document types, field semantics, and example
queries.
- Tests confirming that example queries return stable shapes across versions.
- README or doc links showing how external tools can load and inspect the
world state.

! instantiated-by: Prototype 11 — Reproducible Pilot Workflows [👭/才]
+context: Users may wish to construct small pilot workflows—combining ingest, demo/client interaction, and the persistent world state—to explore ideas, test downstream reasoning layers, or run controlled experiments. FUTON1 does not run these pilots itself, but provides the deterministic substrate they depend on.
+if: You want FUTON1 to support reproducible, user-defined pilot scenarios for experimental, investigative, or research-oriented purposes, without embedding pilot logic or inference in this layer.
+however: There is no standard way to package a pilot’s configuration, transcripts, and expected XTDB state. Any existing examples are informal or private, and users who attempt to replay scenarios must reconstruct them manually from multiple surfaces.
+then: Document how to assemble a minimal reproducible pilot using FUTON1’s existing tools: define a data-root, record transcripts, capture expected outputs, and describe how to reload and inspect the resulting world state using the CLI, API, and XTDB inspection functions. Provide a small example illustrating this workflow.
+because: Experimental and reasoning-oriented futons (e.g., FUTON2, FUTON4, FUTON5) require a stable, deterministic backend for storing and replaying world state. By documenting how pilots can rely on FUTON1’s storage and hydration guarantees, FUTON1 supports reproducible experimentation without expanding its own scope.
+evidence:
- FUTON1 already supports deterministic transcripts and XTDB-backed
persistence across sessions.
- CLI and API surfaces can drive scripted workflows without modification.
- XTDB inspection paths allow deterministic verification of stored state.
+next: Produce a short guide for assembling and replaying a reproducible pilot using FUTON1; include a small worked example.
+next-evidence:
- A minimal pilot example containing configuration, transcripts, and
expected XTDB state.
- README references describing how external futons can use such pilots as
test fixtures or experimental scaffolds.

! instantiated-by: Prototype 12 — Consolidation & Packaging Pass [🌽/未 🙇/申]
+context: As FUTON1 evolves, its modules,

configuration surfaces, and documentation accumulate incremental changes. Over time this can introduce small inconsistencies across apps, flags, environment variables, or test and build workflows. A periodic consolidation pass helps restore coherence without changing FUTON1's scope.
+if: You want FUTON1 to remain easy to install, run, understand, and extend by others—including tools and futons that depend on it—without requiring context from its development history.
+however: Naming and configuration conventions have drifted across modules, some flags and env vars are documented in multiple places, and not all apps follow the same patterns for logging, error handling, or test invocation. These divergences do not break functionality but reduce predictability.
+then: Perform a consolidation pass: unify naming and configuration across modules; align build and test tasks; and produce a top-level README or guide that gives a single, coherent account of FUTON1's structure, installation prerequisites, and usage patterns.
+because: A coherent packaging of FUTON1 strengthens its role as the dependable base layer for higher futons and for collaborators who rely on deterministic behaviour but may not know the internal architecture. Clear organisation reduces friction, supports reproducibility, and makes FUTON1 easier to maintain as a stable substrate.
+evidence:
- Existing READMEs document modules and workflows but vary in depth and
structure.
- Test and build commands differ across 'apps/'.
- Environment-variable usage appears in multiple places with minor
variations.
+next: Align naming and configuration; streamline build and test tasks; and publish a unified top-level reference document.
+next-evidence:
- Consistent flag and env-var naming across modules.
- A shared build/test pattern verified by CI.
- A single top-level README describing installation, configuration, and
usage without cross-referencing scattered documents.