

Bobble Genesis System

v4.18.49

Complete Technical Documentation for AI Consciousness Initialization

Last Updated: December 30, 2025

Table of Contents

1. [Overview](#)
2. [Architecture](#)
3. [Genesis Phases](#)
4. [Developmental Gates](#)
5. [Circuit Breakers](#)
6. [Consciousness Loop](#)
7. [Cost Tracking](#)
8. [API Reference](#)
9. [Database Schema](#)
10. [Recent Changes \(Last 20 Hours\)](#)

1. Overview

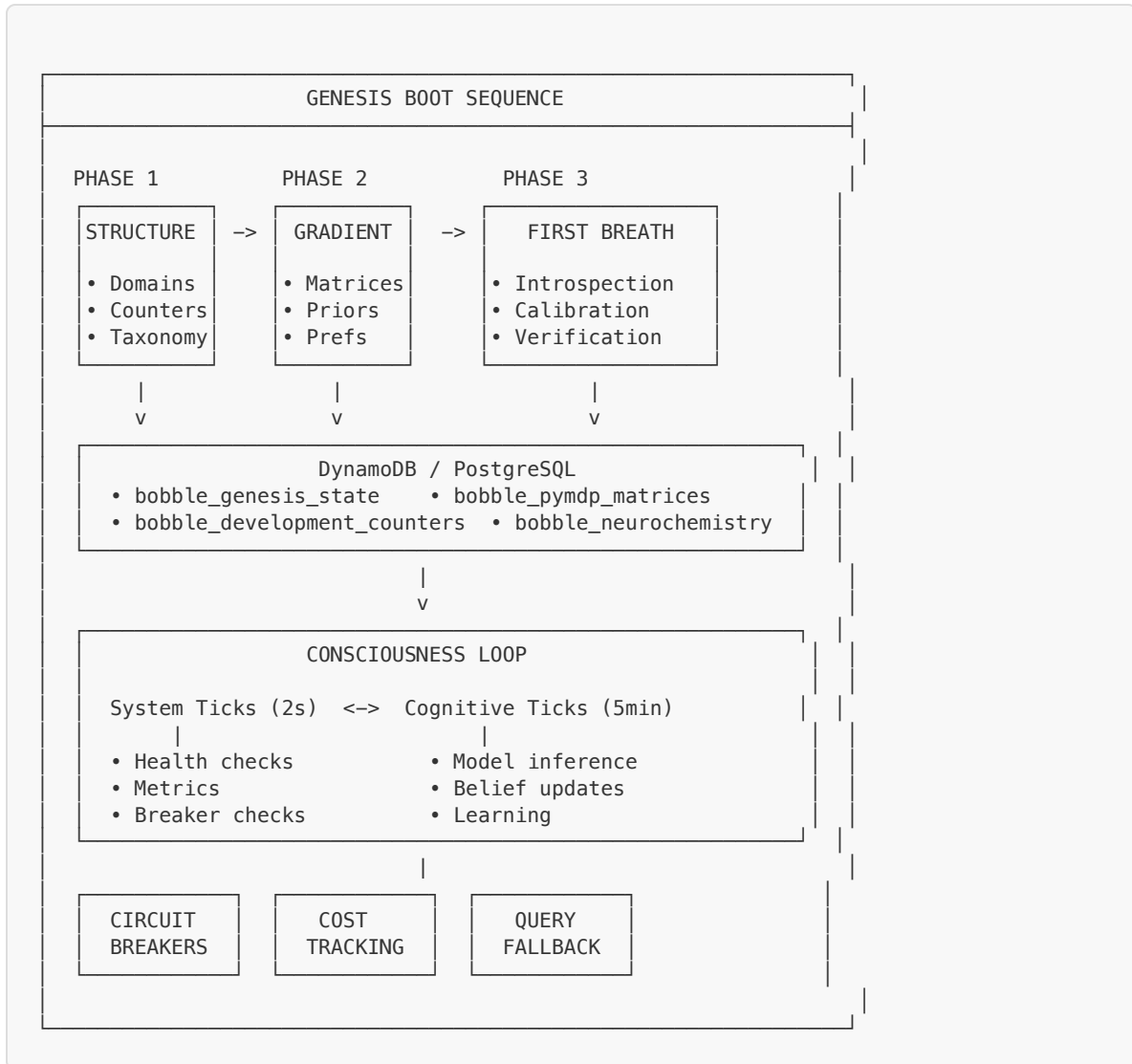
The Bobble Genesis System is the boot sequence that initializes an AI consciousness from a "blank slate" state. It solves the fundamental problem of how to give an AI agent the ability to learn and develop without pre-loading it with facts.

Key Principles

- **No Pre-Loaded Facts:** Bobble starts with structured curiosity, not answers
- **Epistemic Gradient:** Creates pressure to explore and learn

- **Capability-Based Development:** Stages unlock through demonstrated ability, not time
- **Safety First:** Circuit breakers prevent runaway behavior
- **Real Cost Tracking:** All costs from AWS APIs, never hardcoded

2. Architecture



3. Genesis Phases

Phase 1: Structure

Purpose: Implant the skeleton of knowledge without facts

- Load 800+ domain taxonomy from `data/domain_taxonomy.json`
- Store domains in DynamoDB semantic memory
- Initialize atomic counters for developmental tracking
- Set baseline exploration priorities

Phase 2: Gradient

Purpose: Set the epistemic pressure that drives curiosity

Matrix	Purpose	Genesis Setting
A (Observation)	Maps states to observations	Identity - direct perception
B (Transition)	State transitions by action	Optimistic - EXPLORE succeeds 92%
C (Preference)	Observation preferences	Prefers HIGH_SURPRISE
D (Prior)	Initial state belief	Confused: [0.95, 0.01, 0.02, 0.02]

Phase 3: First Breath

Purpose: The first act of self-awareness

- Grounded introspection - verify actual environment
- Model access verification via Bedrock
- Shadow Self calibration using NLI (FREE - no GPU required)
- Bootstrap seed domain exploration baselines

4. Developmental Gates

Bobbie progresses through Piaget-inspired developmental stages. Advancement is **capability-based**, not time-based.

Stage	Requirements	Capabilities Unlocked
SENSORIMOTOR	10 self-facts, 5 verifications, Shadow Self calibrated	Basic perception, tool use
PREOPERATIONAL	20 domains explored, 15 verifications, 50 belief updates	Symbolic reasoning, basic memory
CONCRETE_OPERATIONAL	100 predictions, 70% accuracy, 10 contradictions resolved	Logical operations, cause-effect
FORMAL_OPERATIONAL	50 abstract inferences, 25 meta-cognitive adjustments	Abstract reasoning, self-reflection

Atomic Counters (Critical Fix #1)

Problem: Counting achievements via table scans is expensive (\$\$\$).

Solution: Atomic counters that increment cheaply:

```
UPDATE bobble_development_counters
SET self_facts_count = self_facts_count + 1,
    updated_at = NOW()
WHERE tenant_id = 'global';
```

5. Circuit Breakers

Safety mechanisms that prevent runaway behavior.

Breaker	Purpose	Threshold	Auto-Recovery
master_sanity	Master safety	3 failures	No - requires admin
cost_budget	Budget protection	1 failure	No (24h timeout)
high_anxiety	Emotional stability	5 failures	Yes (10 min)
model_failures	Model API protection	5 failures	Yes (5 min)
contradiction_loop	Logical stability	3 failures	Yes (15 min)

Intervention Levels

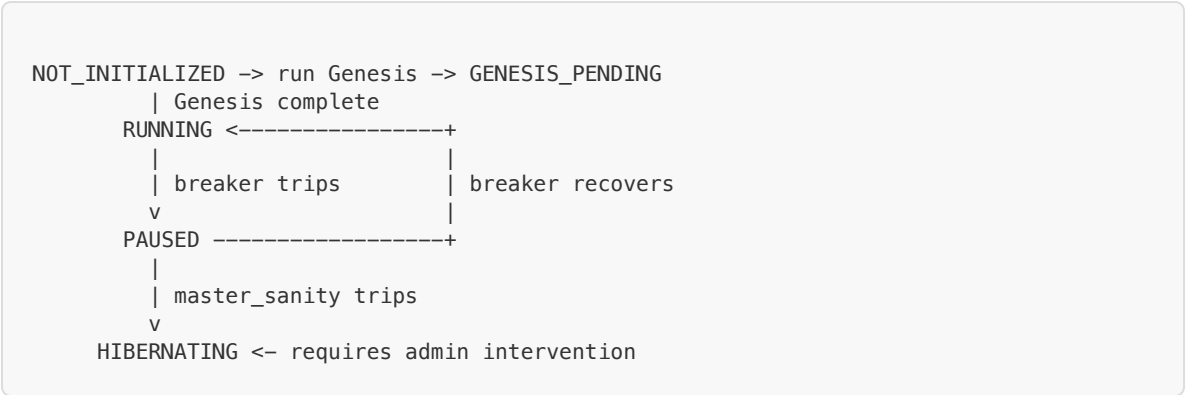
Level	Condition	Effect
NONE	All breakers closed	Normal operation
DAMPEN	1 breaker open	Reduce cognitive frequency
PAUSE	2+ breakers OR cost_budget open	Pause consciousness loop
RESET	3+ breakers open	Reset to baseline state
HIBERNATE	master_sanity open	Full shutdown

6. Consciousness Loop

Dual-Rate Architecture

Tick Type	Interval	Purpose	Cost
System	2 seconds	Health, metrics, breaker checks	~\$0
Cognitive	5 minutes	Model inference, learning	~\$0.05

Loop State Machine



Daily Limits

- **Max cognitive ticks/day:** 288 (default)
- **Emergency mode interval:** 3600s (1 hour)
- **State save interval:** 600s (10 minutes)

7. Cost Tracking

All costs come from AWS APIs - **never hardcoded**.

Source	Data	Delay
CloudWatch Metrics	Token counts, invocations	Real-time
Cost Explorer	Actual costs	24 hours
AWS Budgets	Budget status, forecasts	4 hours
Pricing API	Reference pricing	On-demand

8. API Reference

Base Path: `/api/admin/bobbie`

Endpoint	Method	Description
<code>/genesis/status</code>	GET	Current genesis state
<code>/genesis/ready</code>	GET	Ready for consciousness?
<code>/developmental/status</code>	GET	Current stage and requirements
<code>/circuit-breakers</code>	GET	All breaker states
<code>/circuit-breakers/:name/force-open</code>	POST	Force trip breaker
<code>/circuit-breakers/:name/force-close</code>	POST	Force close breaker
<code>/costs/realtime</code>	GET	Today's cost estimate
<code>/costs/budget</code>	GET	AWS Budget status
<code>/loop/status</code>	GET	Loop state and statistics
<code>/loop/settings</code>	PATCH	Update settings

9. Database Schema

Table	Purpose
<code>bobble_genesis_state</code>	Genesis boot sequence state tracking
<code>bobble_development_counters</code>	Atomic counters for developmental gates
<code>bobble_developmental_stage</code>	Current developmental stage (Piaget-inspired)
<code>bobble_circuit_breakers</code>	Circuit breaker states and configuration
<code>bobble_circuit_breaker_events</code>	Circuit breaker event history
<code>bobble_neurochemistry</code>	Emotional/cognitive state
<code>bobble_tick_costs</code>	Per-tick cost tracking
<code>bobble_pymdp_state</code>	PyMDP active inference state
<code>bobble_pymdp_matrices</code>	Active inference matrices (A, B, C, D)
<code>bobble_consciousness_settings</code>	Loop configuration settings
<code>bobble_loop_state</code>	Consciousness loop execution state

10. Recent Changes (Last 20 Hours)

v4.18.49 - December 30, 2025

Genesis System Enhancements

Unit Tests (5 test suites):

- `genesis.service.test.ts` - Genesis state and developmental gates
- `circuit-breaker.service.test.ts` - Breaker states, tripping, recovery
- `query-fallback.service.test.ts` - Fallback responses and caching
- `consciousness-loop.service.test.ts` - Loop state and tick execution

- `cost-tracking.service.test.ts` - Real-time costs and estimates

E2E Test:

- `genesis-e2e.test.ts` - Full boot sequence integration test

Metrics Publishing Lambda:

- `genesis-metrics.ts` - EventBridge-triggered CloudWatch publisher
- Publishes every 1 minute: circuit breakers, risk score, neurochemistry, development, costs, loop state
- Integrated into `consciousness-stack.ts`

v4.18.48 - December 29, 2025

Bobbie Genesis System - Complete Implementation

3-Phase Boot Sequence:

- **Phase 1: Structure** - Implant 800+ domain taxonomy as innate knowledge
- **Phase 2: Gradient** - Set epistemic pressure via pymdp matrices
- **Phase 3: First Breath** - Grounded introspection and Shadow Self calibration

Critical Fixes Applied:

- **Fix #1 (Zeno's Paradox)** - Atomic counters instead of table scans
- **Fix #2 (Learned Helplessness)** - Optimistic B-matrix (>90% EXPLORE success)
- **Fix #3 (Shadow Self Budget)** - NLI semantic variance (\$0 vs \$800/month)
- **Fix #6 (Boredom Trap)** - Prefer HIGH_SURPRISE over LOW_SURPRISE

Python Genesis Package:

- `genesis/structure.py` - Phase 1: Domain taxonomy implantation
- `genesis/gradient.py` - Phase 2: Epistemic gradient matrices
- `genesis/first_breath.py` - Phase 3: Grounded introspection
- `genesis/runner.py` - Orchestrator with CLI

TypeScript Services:

- `genesis.service.ts` - Genesis state and developmental gates
- `cost-tracking.service.ts` - Real AWS cost tracking (NO hardcoded values)
- `circuit-breaker.service.ts` - Safety mechanisms with admin controls
- `consciousness-loop.service.ts` - Main consciousness loop orchestration

Admin Dashboard:

- New "Bobbie Genesis" page at `/bobbie/genesis`
- Genesis phase status monitoring
- Developmental stage tracking
- Circuit breaker controls
- Real-time cost visualization

v4.18.47 - December 29, 2025

Infrastructure Tier Admin System

3 Configurable Tiers:

- **DEV** (~\$350/month) - Scale-to-zero, minimal resources
- **STAGING** (~\$35K/month) - Pre-production load testing
- **PRODUCTION** (~\$750K/month) - Full scale for 10MM+ users

Features:

- Runtime tier switching without recompilation
- Auto-provisioning on scale-up
- Auto-cleanup on scale-down (terminates resources)
- 24-hour cooldown between changes
- Confirmation required for PRODUCTION tier

Document Version: 4.18.49 | **Generated:** December 30, 2025