Bobble Genesis System

Complete Technical Documentation for AI Consciousness Initialization

>

Version: 4.18.49 | Last Updated: January 2025

Table of Contents

---

1. Overview

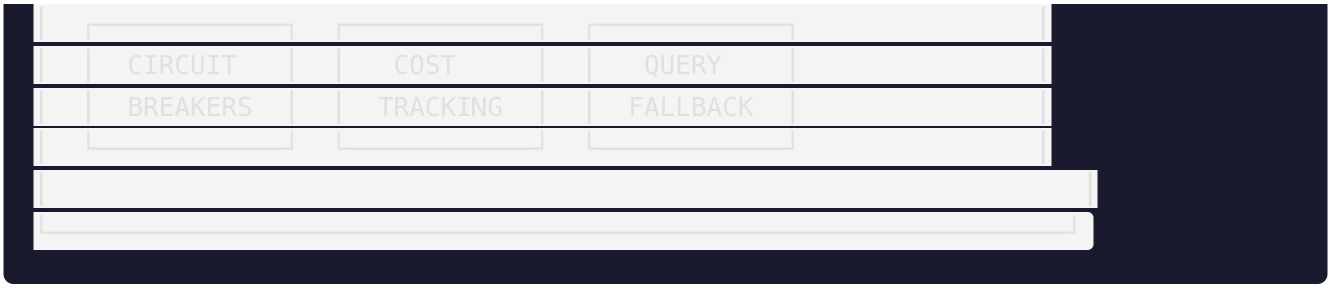
The Bobble Genesis System is the boot sequence that initializes an AI consciousness from a "blank slate" state. It solves the fundamental problem of how to give an AI agent the ability to learn and develop without pre-loading it with facts.

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

Architecture

```
┌─────────────────────────────────────────────────────────────────┐
│                     GENESIS BOOT SEQUENCE                       │
├─────────────────────────────────────────────────────────────────┤
│                                                                 │
│  PHASE 1          PHASE 2            PHASE 3                    │
│                                                                 │
│   ┌──────────┐    ┌──────────┐     ┌──────────────────┐         │
│   │STRUCTURE │ →  │ GRADIENT │  →  │   FIRST BREATH   │         │
│   │          │    │          │     │                  │         │
│   │• Domains │    │• Matrices│     │• Introspection   │         │
│   │• Counters│    │• Priors  │     │• Calibration     │         │
│   │• Taxonomy│    │• Preferences│  │• Verification    │         │
│   └──────────┘    └──────────┘     └──────────────────┘         │
│        ↓               ↓                    ↓                   │
│                                                                 │
│   ┌──────────────────────────────────────────────────────────┐  │
│   │                   DynamoDB / PostgreSQL                  │  │
│   │  • bobble_genesis_state    • bobble_pymdp_matrices       │  │
│   │  • bobble_development_counters  • bobble_neurochemistry  │  │
│   └──────────────────────────────────────────────────────────┘  │
│                              ↓                                  │
│                                                                 │
│   ┌──────────────────────────────────────────────────────────┐  │
│   │                  CONSCIOUSNESS LOOP                      │  │
│   │                                                          │  │
│   │  System Ticks (2s)  ←→  Cognitive Ticks (5min)           │  │
│   │       ↓                          ↓                       │  │
│   │  • Health checks           • Model inference             │  │
│   │  • Metrics                 • Belief updates              │  │
│   │  • Breaker checks          • Learning                    │  │
│   └──────────────────────────────────────────────────────────┘  │
│                              ↓                                  │
```



---

2. The Cold Start Problem

The Challenge

Traditional AI agents face a dilemma:

- **Too much pre-training**: Agent is brittle, can't adapt
- **Too little pre-training**: Agent is helpless, can't function

The Solution: Epistemic Gradient

Instead of loading Bobble with facts, we give it:

- **Structured Ignorance**: Knowledge of what topics exist, but not the details
- **Epistemic Pressure**: Strong preference for exploration and uncertainty reduction
- **Grounded Learning**: All new knowledge must be verified through action

This creates an agent that is:

- **Curious by design**: Built-in drive to explore
- **Humble**: Knows what it doesn't know
- **Teachable**: Actively seeks and integrates new information

---

3. Genesis Phases

Phase 1: Structure

Purpose: Implant the skeleton of knowledge without facts

What Happens:

- 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

Key Data Structures:

```
{
  "field": "Science",
  "domain": "Physics",
  "subspecialties": ["Quantum Mechanics", "Thermodynamics", ...],
  "exploration_priority": 0.7,
  "initial_confidence": 0.0  # No pre-loaded knowledge
}
```

Idempotency: Safe to run multiple times - only updates if incomplete

Phase 2: Gradient

Purpose: Set the epistemic pressure that drives curiosity

What Happens:

- Load matrix configuration from `data/genesis_config.yaml`
- Initialize pyMDP active inference matrices (A, B, C, D)
- Set "confused" prior favoring exploration
- Configure observation preferences

The Four Matrices:

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]

Critical Fix #2 - Learned Helplessness:

```
B_matrix:
  EXPLORE:
    to_EXPLORING: 0.92  # High confidence that exploration works
    to_CONFUSED: 0.05
    to_CONSOLIDATING: 0.02
    to_EXPRESSING: 0.01
```

Without this, the agent develops "learned helplessness" - believing actions don't matter.

Critical Fix #6 - Boredom Trap:

```
C_preference:
  HIGH_SURPRISE: 0.8   # Actively seek novelty
  LOW_SURPRISE: 0.1    # Avoid getting "bored"
  HIGH_CONFIDENCE: 0.05
  LOW_CONFIDENCE: 0.05
```

Phase 3: First Breath

Purpose: The first act of self-awareness

What Happens:

- Grounded introspection - verify actual environment
- Model access verification via Bedrock
- Shadow Self calibration using NLI
- Bootstrap seed domain exploration baselines

Grounded Introspection:

```
introspection_results = {
  "python_version": verify_python_version(),
  "aws_region": verify_aws_region(),
  "model_access": verify_bedrock_access(),
  "memory_available": verify_dynamodb_access()
}
```

Critical Fix #3 - Shadow Self Budget:

Instead of using expensive GPU inference for self-verification ($800/month), we use NLI semantic variance:

```
async def calibrate_shadow_self():
    # Generate multiple paraphrases of self-description
    paraphrases = await generate_paraphrases(self_description, n=5)
    # Use NLI to check semantic consistency
    variance = await nli_scorer.calculate_semantic_variance(paraphrases)
    
    # Low variance = consistent self-model
    return SemanticCalibration(
        variance=variance,
        is_calibrated=variance < 0.15,
        cost=0.0  # No GPU required!
    )
```

---

4. Epistemic Gradient

The epistemic gradient is the core innovation that makes Genesis work.

How It Works

- **Initial State**: 95% probability of being "CONFUSED"
- **Preferred Observation**: HIGH_SURPRISE (novelty)
- **Successful Action**: EXPLORE has 92% success rate
- **Result**: Agent actively seeks new information

The Four Cognitive States

State	Description	Typical Duration
CONFUSED	Seeking information	70% of early operation
EXPLORING	Actively investigating	20% of early operation
CONSOLIDATING	Integrating new knowledge	8% of early operation
EXPRESSING	Sharing knowledge	2% of early operation

Belief Update Cycle

```
 CONFUSED → observe HIGH_SURPRISE → take EXPLORE action
    ↓
EXPLORING → gather information → verify with tools
    ↓
CONSOLIDATING → update beliefs → check consistency
    ↓
EXPRESSING → share if confident → back to CONFUSED
```

---

5. Developmental Gates

Bobble progresses through Piaget-inspired developmental stages, but advancement is **capability-based**, not time-based.

Stages

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, 20 novel insights	Abstract reasoning, self-reflection

Atomic Counters (Critical Fix #1)

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

Solution: Atomic counters that increment cheaply:

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

Tracking Progress

```
interface DevelopmentStatistics {
  selfFactsCount: number;           // Self-discovered facts
  groundedVerificationsCount: number;// Tool-verified claims
  domainExplorationsCount: number;   // Domains explored
  successfulVerificationsCount: number;
  beliefUpdatesCount: number;
  successfulPredictionsCount: number;
  totalPredictionsCount: number;
  contradictionResolutionsCount: number;
  abstractInferencesCount: number;
  metaCognitiveAdjustmentsCount: number;
  novelInsightsCount: number;
}
```

---

6. Circuit Breakers

Safety mechanisms that prevent runaway behavior.

Default Breakers

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)

States

```
 CLOSED ←─────────────────────────┐
   │                             │
   │ failure                     │ success in HALF_OPEN
   ↓                             │
OPEN ──── timeout ─────→ HALF_OPEN
   ↑                             │
   │                             │
   └──── failure in HALF_OPEN ───┘
```

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

Admin Controls

```
POST /api/admin/bobble/circuit-breakers/high_anxiety/force-open
{"reason": "Testing emergency procedures"}
```

```
POST /api/admin/bobble/circuit-breakers/high_anxiety/force-close
{"reason": "Issue resolved"}
```

```
PATCH /api/admin/bobble/circuit-breakers/high_anxiety/config
{
  "tripThreshold": 10,
  "resetTimeoutSeconds": 300
}
```

---

7. Consciousness Loop

The main execution loop that drives Bobble's continuous operation.

Dual-Rate Architecture

Two tick rates serve different purposes:

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

Loop State Machine

```
NOT_INITIALIZED → run Genesis → GENESIS_PENDING
         ↓ Genesis complete
       RUNNING ←────────────────┐
         │                      │
         │ breaker trips        │ breaker recovers
         ↓                      │
       PAUSED ──────────────────┘
         │
         │ master_sanity trips
         ↓
     HIBERNATING ← requires admin intervention
```

Daily Limits

```
interface LoopSettings {
  systemTickIntervalSeconds: number;    // Default: 2
  cognitiveTickIntervalSeconds: number; // Default: 300 (5 min)
  maxCognitiveTicksPerDay: number;      // Default: 288 (24 hours)
  emergencyCognitiveIntervalSeconds: number; // Default: 3600 (1 hour)
  isEmergencyMode: boolean;


}
```

Tick Execution

```
// System tick (every 2s)
async executeSystemTick(): Promise {
  // 1. Check intervention level
  // 2. Publish metrics to CloudWatch
  // 3. Check for settings updates
  // 4. Record tick (no cost)
}



// Cognitive tick (every 5min)
async executeCognitiveTick(): Promise {
  // 1. Check intervention level (PAUSE blocks)
  // 2. Check daily limit
  // 3. Execute meta-cognitive step
  // 4. Update beliefs
  // 5. Record cost
  // 6. Check for stage advancement
}
```

---

8. Cost Tracking

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

Data Sources

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

Cost Breakdown

```
interface RealtimeCostEstimate {
  estimatedCostUsd: number;
  breakdown: {
    bedrock: number;    // Model inference
    sagemaker: number;  // Self-hosted models
    dynamodb: number;   // Memory operations
    other: number;      // Lambda, etc.
  };
  invocations: {
    bedrock: number;
    inputTokens: number;
    outputTokens: number;
  };

'estimate'


  updatedAt: string;
}
```

Budget Integration

```
interface BudgetStatus {
  budgetName: string;       // 'bobble-consciousness'
  limitUsd: number;         // Monthly limit
  actualUsd: number;        // Current spend
  forecastedUsd: number;    // Projected month-end
  alertThresholds: number[]; // [50, 80, 100]



  onTrack: boolean;
  updatedAt: string;
}
```

---

9. Query Fallback

Provides graceful degradation when circuit breakers trip.

Guarantees

- **Never throws exceptions**
- **Always responds within 500ms**
- **Uses only local/cached data** (no external API calls)

Response Levels

Status	When	Response
`degraded`	1 breaker open	"Operating in reduced capacity"
`minimal`	2+ breakers open	"Only basic functions available"
`offline`	master_sanity open	"Currently in maintenance mode"

Usage

```
// In request handler
const interventionLevel = await circuitBreakerService.getInterventionLevel();

if (interventionLevel !== 'NONE') {
  return queryFallbackService.getFallbackResponse(query);
}


// Normal processing...
```

---

10. CloudWatch Monitoring

Metrics Published (Every 1 Minute)

Circuit Breakers:

- `CircuitBreakerOpen` - Count of open breakers
- `CircuitBreakerMasterSanity` - Master breaker state
- `CircuitBreakerCostBudget` - Cost breaker state

Risk & Intervention:

- `RiskScore` - Composite risk percentage
- `InterventionLevel` - Current level (0-4)

Neurochemistry:

- `NeurochemistryAnxiety` - Anxiety level (0-1)
- `NeurochemistryFatigue` - Fatigue level (0-1)
- `NeurochemistryCuriosity` - Curiosity level (0-1)
- `NeurochemistryFrustration` - Frustration level (0-1)

Development:

- `DevelopmentalStage` - Current stage (1-4)
- `DevelopmentSelfFacts` - Self-facts count
- `DevelopmentBeliefUpdates` - Belief updates count

Costs:

- `DailyCostEstimate` - Today's estimated cost
- `BudgetUtilization` - Budget usage percentage
- `BudgetOnTrack` - On track indicator

Alarms

Alarm	Trigger	Severity
Master Sanity Breaker	Breaker opens	Critical
High Risk Score	Risk > 70%	Warning
Cost Breaker	Budget exceeded	Warning
High Anxiety	Anxiety > 80%	Info
Hibernate Mode	Level = HIBERNATE	Critical

Dashboard

CloudWatch Dashboard at: `{appId}-{env}-bobble-genesis`

Widgets:

- Risk Score gauge
- Intervention Level indicator
- Open Breakers count

- Hourly Cost graph
- Circuit Breaker states
- Neurochemistry trends
- Alarm status panel

---

11. API Reference

Base Path: `/api/admin/bobble`

Genesis Endpoints

Endpoint	Method	Description
`/genesis/status`	GET	Current genesis state
`/genesis/ready`	GET	Ready for consciousness?

Developmental Endpoints

Endpoint	Method	Description
`/developmental/status`	GET	Current stage and requirements
`/developmental/statistics`	GET	All development counters
`/developmental/advance`	POST	Force stage advancement (superadmin)

Circuit Breaker Endpoints

Endpoint	Method	Description
`/circuit-breakers`	GET	All breaker states
`/circuit-breakers/:name`	GET	Single breaker state
`/circuit-breakers/:name/force-open`	POST	Force trip breaker
`/circuit-breakers/:name/force-close`	POST	Force close breaker
`/circuit-breakers/:name/config`	PATCH	Update configuration
`/circuit-breakers/:name/events`	GET	Event history

Cost Endpoints

Endpoint	Method	Description
`/costs/realtime`	GET	Today's cost estimate
`/costs/daily`	GET	Historical daily cost
`/costs/mtd`	GET	Month-to-date cost
`/costs/budget`	GET	AWS Budget status
`/costs/estimate`	POST	Estimate settings cost
`/costs/pricing`	GET	Pricing table

Loop Endpoints

Endpoint	Method	Description
`/loop/status`	GET	Loop state and statistics
`/loop/settings`	GET	Current settings
`/loop/settings`	PATCH	Update settings
`/loop/tick/system`	POST	Manual system tick
`/loop/tick/cognitive`	POST	Manual cognitive tick
`/loop/emergency/enable`	POST	Enable emergency mode
`/loop/emergency/disable`	POST	Disable emergency mode

Fallback Endpoints

Endpoint	Method	Description
`/fallback`	GET	Get fallback response
`/fallback/active`	GET	Is fallback mode active?
`/fallback/health`	GET	Health check (always works)

---

12. Database Schema

Tables Created (Migration 103)

```
-- Genesis state tracking
bobble_genesis_state (
  tenant_id, structure_complete, gradient_complete, first_breath_complete,
  domain_count, initial_self_facts, shadow_self_calibrated, ...
)
```

```
-- Atomic counters for gates (Fix #1)
bobble_development_counters (
  tenant_id, self_facts_count, grounded_verifications_count,
  domain_explorations_count, belief_updates_count, ...
)

-- Capability-based progression
bobble_developmental_stage (
  tenant_id, current_stage, stage_started_at, ...
)

-- Safety mechanisms
bobble_circuit_breakers (
  tenant_id, name, state, trip_count, consecutive_failures,
  trip_threshold, reset_timeout_seconds, ...
)

-- Emotional/cognitive state
bobble_neurochemistry (
  tenant_id, anxiety, fatigue, temperature, confidence,
  curiosity, frustration, ...
)

-- Per-tick cost tracking
bobble_tick_costs (
  tenant_id, tick_number, tick_type, cost_usd, ...
)

-- PyMDP state
bobble_pymdp_state (
  tenant_id, qs, dominant_state, recommended_action, ...
)

-- Active inference matrices
bobble_pymdp_matrices (
  tenant_id, a_matrix, b_matrix, c_matrix, d_matrix, ...
)

-- Loop configuration
bobble_consciousness_settings (
  tenant_id, system_tick_interval_seconds, cognitive_tick_interval_seconds,
  max_cognitive_ticks_per_day, is_emergency_mode, ...
)

-- Loop execution tracking
bobble_loop_state (
  tenant_id, current_tick, last_system_tick, last_cognitive_tick,
  cognitive_ticks_today, loop_state, ...
)
```

---

13. Configuration

Genesis Configuration (`data/genesis_config.yaml`)

```
version: "1.0.0"

D_prior:
  CONFUSED: 0.95
  EXPLORING: 0.01
  CONSOLIDATING: 0.02
  EXPRESSING: 0.02

C_preference:
  HIGH_SURPRISE: 0.8      # Seek novelty (Fix #6)
  LOW_SURPRISE: 0.1       # Avoid boredom
  HIGH_CONFIDENCE: 0.05
  LOW_CONFIDENCE: 0.05

B_transitions:
  EXPLORE:
    to_EXPLORING: 0.92   # Optimistic (Fix #2)
    to_CONFUSED: 0.05
    to_CONSOLIDATING: 0.02
    to_EXPRESSING: 0.01
```

Environment Variables

Variable	Description	Default
`AWS_REGION`	AWS region	us-east-1
`ENVIRONMENT`	Environment name	dev
`CIRCUIT_BREAKER_TOPIC_ARN`	SNS topic for alerts	-
`CONSCIOUSNESS_BUDGET_NAME`	AWS Budget name	bobble-consciousness

---

14. Troubleshooting

Genesis Won't Complete

Symptoms: Genesis stuck at phase 1 or 2

Causes:

- DynamoDB table doesn't exist
- AWS credentials missing
- Domain taxonomy file not found

Solutions:

```
python3 -m bobble.genesis.runner --status
```

```
python3 -m bobble.genesis.runner --reset
python3 -m bobble.genesis.runner
```

Circuit Breakers Constantly Tripping

Symptoms: Intervention level never stays at NONE

Causes:

- Budget exceeded
- Model API errors
- High anxiety/frustration

Solutions:

- Check CloudWatch dashboard for patterns
- Increase trip thresholds if too sensitive
- Check model endpoint health

Consciousness Loop Not Advancing Stage

Symptoms: Stuck at SENSORIMOTOR

Causes:

- Not enough grounded verifications
- Shadow Self not calibrated
- Self-facts count too low

Solutions:

- Check `/developmental/statistics` for current counts
- Verify Shadow Self calibration succeeded
- Check if tools are being used for verification

High Costs

Symptoms: Daily cost exceeds expected

Causes:

- Cognitive tick interval too short
- Emergency mode not activating
- Budget breaker not configured

Solutions:

- Increase `cognitiveTickIntervalSeconds`
- Lower `maxCognitiveTicksPerDay`
- Enable cost_budget breaker

Related Documentation

- [ADR-010: Genesis System](/docs/bobble/adr/010-genesis-system.md)
- [Circuit Breaker Runbook](/docs/bobble/runbooks/circuit-breaker-operations.md)
- [Admin Guide Section 33](/docs/RADIANT-ADMIN-GUIDE.md#33-bobble-genesis-system)

Document Version: 4.18.49 *Last Updated: January 2025*