Fly Machines Made Micro-VMs Fast. The VMs Still Need External Orchestration.

by Nick Clark | Published March 28, 2026 | PDF

Fly Machines provides fast-starting micro-VMs that can be started, stopped, and migrated through an API. The VM abstraction provides stronger isolation than containers with near-container startup times. But Fly Machines are externally managed: an API starts them, stops them, and configures them. They do not carry their own execution cycle, self-evaluate their state, or autonomously decide when to activate and deactivate. The gap is between fast micro-VM hosting and memory-resident objects that self-execute from their own governed state.

Fly Machines' fast micro-VM technology and global deployment capabilities represent genuine infrastructure innovation. The gap described here is about the execution model.

API-managed lifecycle without self-governance

A Fly Machine is started through an API call, executes until stopped or it exits, and can be stopped and restarted through the API. The lifecycle is externally managed. The machine does not decide when to start or stop based on its own state evaluation. It does what the API tells it to do.

Memory-resident execution objects govern their own lifecycle: waking when conditions are met, entering dormancy when appropriate, and resuming when new information arrives. The lifecycle is self-governed, not API-managed.

Auto-stop and auto-start without semantic evaluation

Fly Machines can auto-stop when idle and auto-start when a request arrives. This is efficient resource usage. But the stop and start decisions are based on request activity, not on semantic evaluation of the machine's governed state. An object that should remain active because its governance requires continuous monitoring would auto-stop due to lack of incoming requests.

What memory-resident execution provides

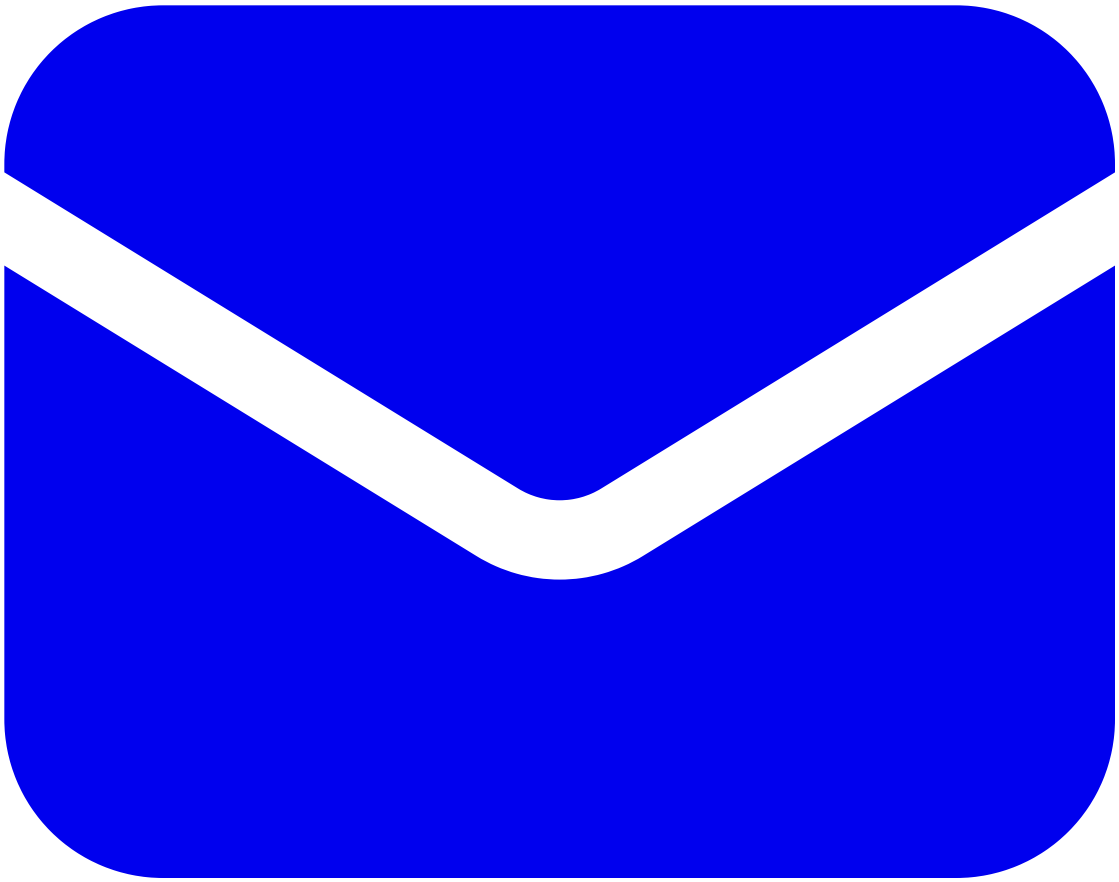
Memory-resident execution objects decide their own activation state based on semantic evaluation of their governed memory. Dormancy is entered when the object determines its task is complete or conditions are not met. Wake is triggered by semantic conditions, not just incoming requests. Fly Machines' fast micro-VM start times could provide the substrate for memory-resident objects that wake and sleep based on their own governed state.

Memory-Resident Execution All 21 steps →

Persistent objects that execute without orchestration.

Patent
US 19/538,221 · filed
Primary Technical Disclosure
○ Memory-Resident Execution: Persistent Semantic Objects Without Orchestration
Secondary Technical
○ Six-Action Execution Evaluation Cycle: Parse, Evaluate, Select at Every Node○ Cognition-Authority-Execution Separation: Reasoning Cannot Authorize Action○ Dormancy as First-Class Execution State: Valid Suspension Without Failure○ Semantic Backoff: Retry Pacing From Execution Outcomes Rather Than Fixed Timers○ Wake Triggers for Dormancy Exit: Explicit Reentry Conditions in Memory○ Persistent Polling Behavior: Autonomous Condition Evaluation Without Schedulers○ Intent Refinement During Execution: Adaptive Objectives Without Re-Instantiation○ Compositional Execution Through Recursive Delegation: Parent-Child Lineage Tracking○ Negative Capability Signals: Recording What Cannot Be Done as Structured Constraint○ Swarm-Based Execution Emergence: Coordinated Behavior Without Centralized Control○ Latency and Failure as Semantic Signals: Structured Inputs From Adverse Conditions○ LLM as Advisory Execution Node: Inference Without Authority Over Agent State○ Append-Only Memory Field: Complete Execution Lineage Through Immutable Records
Applications (General)
○ Serverless Execution Without Cold Starts or State Loss○ Long-Running Autonomous Workflows Without External Orchestration○ Drone Operations Surviving Disconnection○ Deep Space Agent Execution Without Ground Control○ Underwater Robotic Operations Without Connectivity○ Rural Healthcare Agents Surviving Intermittent Connectivity○ Operations in Infrastructure-Destroyed Environments○ Offline Financial Transaction Agents
Applications (Specific)
○ Cloudflare Durable Objects Made State Local. The Objects Still Need Orchestration.○ Azure Service Fabric Actors Are Addressable. They Are Not Autonomous.○ Akka Perfected the Actor Model. Actors Still React Instead of Self-Execute.○ Orleans Made Virtual Actors Practical. The Actors Still Execute on Request.○ Dapr Provides a Sidecar Runtime for Microservices. The Services Still Need External Orchestration.○ wasmCloud Runs WebAssembly Actors. The Actors Wait for Messages.○ Spin Made WebAssembly Serverless. The Functions Are Still Trigger-Based.○ Fermyon Built the WebAssembly Cloud. The Cloud Hosts Functions, Not Self-Executing Objects.● Fly Machines Made Micro-VMs Fast. The VMs Still Need External Orchestration.○ Railway Simplified Application Deployment. The Applications Still Depend on External Execution Triggers.
Memory-Resident Execution overview →
AQ
deterministic
autonomy

Last updated: 2026-03-03



-
- nick@qu3ry.net
- 72 28 14 36 01

Invented by Nick Clark | Founding Investors: Devin Wilkie