DASHECORP AS · OSLO

WHITEPAPER V1.5
MAY 2026
GENERAL EDITION

A whitepaper, and a founder's pitch.

The Rig.

An autonomous engineering fleet under operator governance. Five iOS apps in market. Zero operator keystrokes in production this quarter.

Stig-Johnny Støbakk
FOUNDER, DASHECORP AS

OSLO · 2026

01 OPENER

One operator.
Five apps in market.

This is a whitepaper about a working autonomous engineering platform, and a pitch from the founder who built and runs it. The platform is The Rig. The company is Dashecorp. The bet is that the bottleneck of small software companies, which is engineers doing routine work, has a structural fix.

5	4	*0*	17	1
IOS APPS SHIPPING	AGENT PERSONAS IN PRODUCTION	OPERATOR CODE KEYSTROKES THIS QUARTER	YEARS ENGINEERING EXPERIENCE	OPERATOR RUNNING THE RIG (TODAY)

I spent seventeen years writing code for other people. Startups, consultancies, healthcare infrastructure, distributed teams across Norway and Portugal. The whole time I had the same observation: most engineering hours go on routine work that the engineer could describe in three sentences before doing it. The hard parts, naming the thing, deciding the trade-off, owning the failure, are a fraction of the wall-clock cost.

The Rig is what happens when you take that observation seriously and build the structural fix. It is a fleet of autonomous agents that claim issues, write code, open pull requests, review each other's work, and ship through GitHub to App Store and Cloudflare Pages. Five iOS apps and the web properties around them flow through the same pipeline. The operator, me today, writes the issue and approves the merge. Everything in between is structured handoff between agents.

This document is two things at once. It is an honest technical whitepaper, including the gaps that are not yet right. And it is the pitch from a founder who has stopped contracting and started building a company on the leverage this platform creates. The portfolio is the proof. The rig is the product. The next chapter is what the next round funds.

> *Most software-company cost is engineers doing work an agent could do. The rig is the structure that makes that swap safe.*
>
> THE THESIS · §13

02 THE PROBLEM

Why a copilot is *not enough.*

Code-completion tools save keystrokes. They do not save engineering capacity. A copilot waits for direction. It does not close issues, open pull requests, or iterate on review feedback. The cost of an engineering team is the cost of humans doing work that a well-bounded agent system could do without them.

QUESTION	A COPILOT	THE RIG
Who picks up the issue?	A human reads it, pastes context into the editor.	Conductor-E dispatches it to a Dev-E pod within seconds.
Who writes the code?	The human types; the model autocompletes.	Dev-E reads context, clones the repo, writes code and tests.
Who opens the PR?	The human.	Dev-E, with linked issue, named branch, correct frontmatter.
Who reviews?	Another human, hours or days later.	Review-E, on every PR webhook, with a cron safety net.
Who responds to review?	The original human, when back at the keyboard.	Dev-E. Pushes fixes, resolves threads, reruns tests.
Who merges?	A human, after the back-and-forth.	Auto-merge fires when gates pass. Humans merge only T3.
Where is the human?	In the loop on every step.	At the boundaries: intent in, irreversible decisions out.

THREE PROBLEMS A COPILOT DOES NOT HAVE TO SOLVE

Assignment exclusivity. If two agents pick up the same issue, they fork work and produce conflicting PRs. The rig guarantees exactly-once delivery through an event-sourced dispatch layer. A second concurrent caller for the same issue gets a different one, or a 204.

Acceptance gating. Agent output must pass human-authored quality checks before it lands. Agents cannot approve their own work without erasing the gate that makes review meaningful. Review-E is scoped to never review PRs she authored; branch protection and CODEOWNERS are the human veto layer.

Failure recovery. Agents get stuck. Systems go down. A production rig must detect, escalate, and recover without a human polling for problems. StuckGuard catches no-progress loops; the conductor watches lifecycle events; unhandled escalations route to a single admin channel in Discord. Each one of these bit in production before it became a structural rule.

03 THE AGENTS

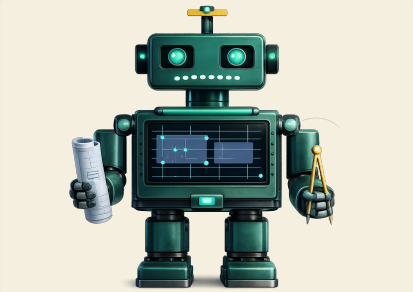
Meet the *engineering team.*

Five personas. Each has a job, a model, a budget, and a set of things it is not allowed to do. They were not designed as a metaphor. They were designed to make the boundaries enforceable. The portraits are how the company recognises them; the contracts are how the platform does.

Conductor-E

ORCHESTRATOR

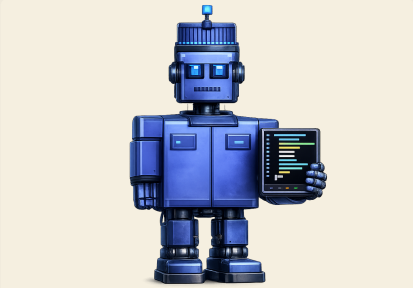
C# · Marten · no LLM



Planner-E

INTAKE

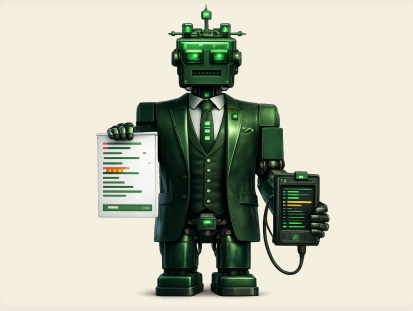
Discord → Issues · Opus 4.7



Dev-E

IMPLEMENTATION

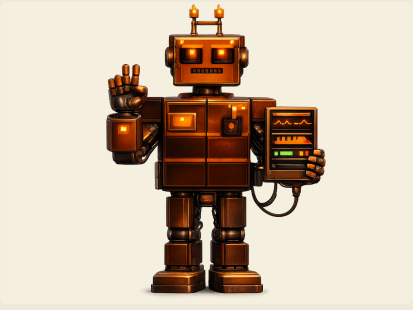
Node · .NET · Sonnet 4.6



Review-E

ACCEPTANCE

PR webhook · Opus 4.7



iBuild-E

IOS / MACOS

Mac Mini · Opus 4.7

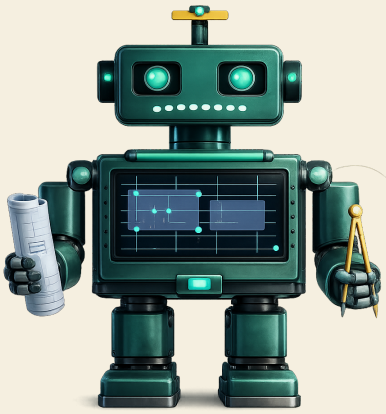
PERSONA · THE ORCHESTRATOR

Conductor-E

Owns the pipeline. Receives every GitHub webhook, classifies the event, decides what happens next. Dispatches work, enforces gates, performs server-side merges of pull requests that pass every check.

Conductor-E is the only agent in the rig with no LLM at all. The orchestration logic is deterministic on purpose: assignment exclusivity, state transitions, and merge gates are too consequential to delegate to a model. The reasoning lives elsewhere; the discipline lives here.

`Deployment rig-conductor · runtime C# · event store Marten on Postgres · merge gate server-side · escalation channel #admin`



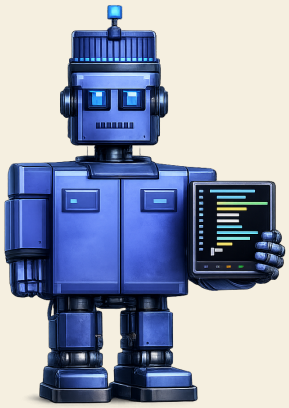
PERSONA · THE INTAKE

Planner-E

Holds a Discord conversation, asks the clarifying questions, converges on an epic plus a tree of child issues with task specs and tier labels. The persona that closes the input-side bottleneck by turning a human sentence into a structured plan the rest of the fleet can claim.

Discord webhook scales the deployment from zero to one when a thread opens. The model is Opus 4.7, temperature 0.1, multi-provider fallback. Planning sessions are long and conversational, and the decomposition quality compounds through every downstream agent. The highest-reasoning tier is the right tier for the work.

`Deployment rig-planner · scale-to-zero via KEDA · multi-provider · status in production, Q3 hardening`



PERSONA · THE IMPLEMENTATION

Dev-E

Claims an issue atomically, clones the repository, reads the project brief, writes code and tests, opens a pull request, and iterates on review feedback until the gates pass. The agent that does what an engineer does after the meeting.

Polyglot by design. Node and .NET active in production, Python paused. Sonnet 4.6 with multi-provider fallback through both claude-cli and codex-cli. Long context, long sessions, careful prompt caching to keep token costs under the per-task ceiling.

`Runtimes Node · .NET · Python (paused) · per-task cap $2 · per-session $15 · provider fallback claude → codex`

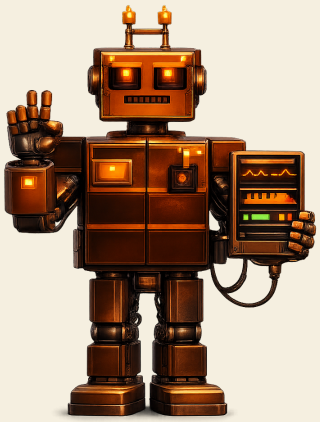
PERSONA · THE ACCEPTANCE GATE

Review-E

Reads every diff against its issue and against the memory store. Approves PRs that meet the contract. Files structured change requests, with line-level comments, when they do not. Never reviews her own work. A structural rule, not a guideline.

Runs Opus 4.7, the highest-reasoning model in the fleet. Review quality is the place not to economise: every gate downstream is only as honest as the gate before it. Webhook-driven on every PR event, with a cron safety net for missed deliveries.

`Model Opus 4.7 · dispatch PR webhook + cron · per-task cap $1.50 · scope never reviews own PRs`



PERSONA · THE IOS / MACOS BUILD

iBuild-E

Builds, signs, and submits the five-app iOS portfolio. Lives on a Mac Mini in Oslo, on the Tailscale mesh, because Xcode requires macOS hardware and the rig is honest about it. The production runtime for everything that ends up in the App Store or TestFlight.

Same model tier as Review-E. Opus 4.7. Build sessions are substantial: Xcode invocations, code signing, App Store Connect submissions, TestFlight pushes. The cost ceiling per session is the highest in the fleet, and the work earns it.

`Host Mac Mini · Oslo · mesh Tailscale · model Opus 4.7 · destinations App Store · TestFlight`

NAMING · HOW THE BRAND AND THE DEPLOYMENT RELATE

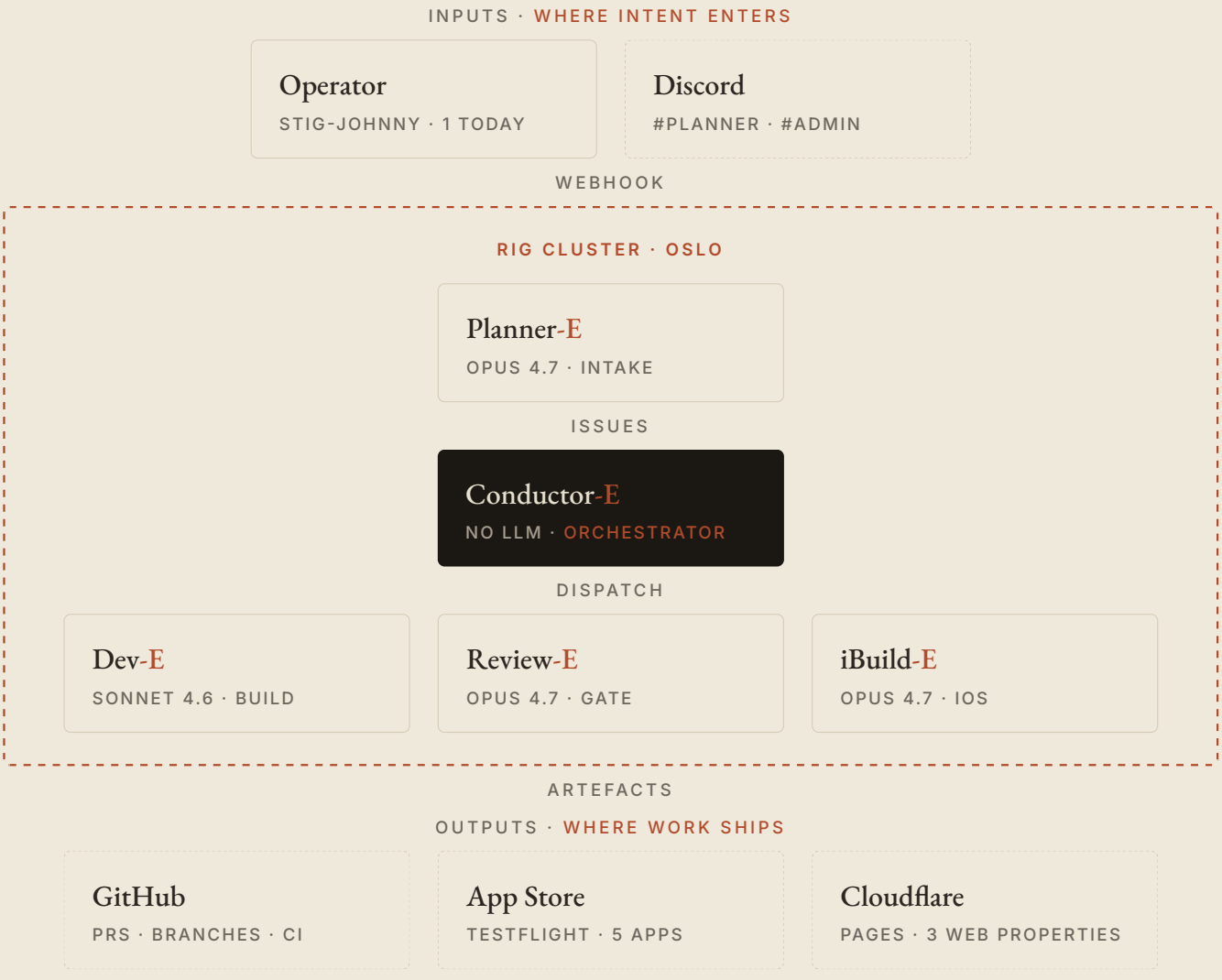
The suffix -E denotes a persona. Conductor-E, Dev-E, Review-E, iBuild-E, Planner-E. Some Dashe Corp apps share the convention: Nutri-E, Cuti-E, Reward-E, Fast-E, Count-E. It is a deliberate brand line, where the apps and the agents that built them carry the same family mark.

The prefix rig- denotes a cluster deployment. rig-conductor, rig-memory-mcp, rig-agent-runtime, rig-gitops. Some entities are both: Conductor-E runs in rig-conductor; Planner-E in rig-planner. The persona is the face of the work; the deployment is the kubernetes thing that scales it.

04 ARCHITECTURE

How the rig is *wired.*

The whole system on one page. Intent enters through a Discord thread at the top. Five agents in a Kubernetes cluster in Oslo do the work in the middle. Code, builds, and deploys leave through GitHub, the App Store, and Cloudflare at the bottom. Everything between is structured handoff.



05 HOW A FEATURE SHIPS

From a Discord thread to a *deployed release.*

Six steps. Each has one owner. Each handoff is structured. The only human touches are at the boundaries: the original intent, and the merge of irreversible changes. Every iOS feature, every marketing page, every infrastructure change at Dashecorp flows through this same pipeline.

Plan. The operator opens a thread in #planner with what they want. Planner-E asks clarifying questions, holds state until the plan is clear, files structured issues with task specs and tier labels. No issues land until the conversation converges.

Build. Dev-E polls for assignments, claims atomically through the dispatcher, clones the repo, reads BRAIN.md and AGENTS.md plus the issue body, writes code and tests, pushes a feature branch named by convention.

PR. Dev-E opens a pull request with `Closes #N` in the body, posts progress to the Discord thread, includes the right frontmatter so downstream automation can read it.

Review. Conductor-E dispatches Review-E on every PR webhook, with a cron safety net for missed events. Review-E reads diff plus issue plus memory and either approves or files `CHANGES_REQUESTED` with line-level comments.

Merge. When CI is green, Review-E approved, all threads resolved, and `Closes #N` is present, auto-merge fires for T0–T2. T3 pull requests require an explicit human signature, enforced at GitHub branch protection.

Deploy. Cloudflare Pages reads the new commit and deploys within seconds for web. Xcode Cloud detects iOS tags and ships to TestFlight. The operator gets a Discord notification with the deployed URL.

06 TRUST MODEL

Autonomy by *blast radius.*

Not every change is equally risky. Editing documentation is reversible. Dropping a database table is not. The rig grades every issue by how reversible it is, and only the safest tiers run fully autonomously. The dangerous tier always requires a human signature, even if the agent has a perfect record. Irreversibility is a structural reason, not a trust metric.

T0

NON-BLAST

Trivially reversible. Cannot affect production behaviour. Documentation edits, test additions, YAML lint fixes, GitHub Actions formatting.

`Full agent autonomy. CI plus Review-E.`

T1

CONTAINED

Reversible within minutes via feature flag or rollback. Single-service feature, test-covered refactor, bounded UI change, dependency bump.

`Full agent under canary. CI + Review-E + SLO gate + error-budget check.`

T2

CROSS-CUTTING

Slow to roll back. Spans multiple repos or services. Event schema change, new public API surface, multi-repo work, shared library bump.

`Agent plans, human approves interface, agent implements. Kyverno two-attestor.`

T3

IRREVERSIBLE

Data loss, security regression, or compliance impact possible. Destructive DB migration, auth code, payment processing, secret rotation.

`Human drives, agent assists. Kyverno rejects admission without human OIDC attestation.`

PROMOTION IS EARNED

T0 to T1 after twenty consecutive successful T0 runs of that task class, with zero human-rework and zero rollbacks. T1 to T2 after twenty successful T1 runs, with zero canary aborts and zero SLO-budget depletions. T2 to T3 has no automatic path: every T3 instance gets a human co-sign regardless of track record.

DEMOTION IS IMMEDIATE

Any attributable rollback drops the ceiling one tier with a thirty-day cooldown. A model version change resets all ceilings to T0 pending human review. The point is to make autonomy expensive to gain and cheap to lose.

> **THREE-LAYER ENFORCEMENT · DEFENCE IN DEPTH**
>
> Layer 1: the dispatch filter refuses to give a T3 issue to an agent. Layer 2: Review-E refuses to approve a T3 PR. Layer 3: Kyverno rejects T3 manifests without a human Sigstore co-sign. A failure in one layer does not compromise the others.

07 THE PORTFOLIO

The portfolio is *the proof.*

The rig is not a demo. It is the engineering team for Dashecorp. The output is a portfolio of shipping software across three surfaces: iOS apps on the App Store, web properties in production, and the rig's own infrastructure running in a Kubernetes cluster in Oslo. None of it requires a human to write production code.

IOS · FIVE APPS IN MARKET

APP	WHAT IT IS	WHERE	STATUS
Nutri-E	AI-powered supplement tracker. Helps users log and analyse what they take.	nutrieapp.com	APP STORE
Fast-E	Intermittent fasting timer. Fast cycles, reminders, history.	dashecorp.com/fast-e	APP STORE
Count-E	Daily counter and date tracker. Anything you want to count or measure between dates.	dashecorp.com/count-e	APP STORE
Reward-E	Star rewards for kids. Parents set goals, kids earn stars, families track progress.	apps.apple.com/app/reward-e	APP STORE
Cuti-E	Character-driven app feedback. Turns user feedback into actionable signal.	cuti-e.com	LAUNCHING

WEB · PRODUCTION PROPERTIES

dashecorp.com	Corporate site. Apps portfolio, the rig overview, the Codi-E build log, per-app legal and support pages.
nutrieapp.com	Nutri-E marketing site. Features, screenshots, App Store download.
cuti-e.com	Cuti-E marketing and documentation. Pre-launch sign-ups and developer docs.

ONE PIPELINE, THREE SURFACES

Consumer iOS work, Dashecorp web work, and rig infrastructure work all flow through the same pipeline. The operator's job is the same regardless of which surface the issue touches. This page was very likely typed by an agent that runs in the cluster it describes.

08 ECONOMICS

Provider-agnostic. Mode-agnostic.

The rig works with any LLM provider, on any billing mode. Subscription tiers like Claude Max for internal use; metered API for partner deployments. Switching is a config change, not a rewrite.

Internal use today runs on subscription tiers, Claude Max plus Codex CLI, fixed monthly, well under the equivalent metered spend for the current load. The dashboard's metered-equivalent figure is a ceiling for internal cost management and a pricing input for partner deployments tomorrow.

Partner deployments run on metered API. That scales with the partner's load, keeps costs attributable per deal, and aligns the unit economics. Same architecture, different billing mode. The provider abstraction makes the swap a configuration commit, not a rewrite.

PER-AGENT CAPS · METERED-EQUIVALENT · ENFORCED AT THE PROXY

AGENT	MODEL	PER TASK	PER SESSION	REASON FOR THE TIER
Planner-E	Opus 4.7	$2	$15	Multi-turn decomposition where planning quality compounds downstream. Highest-reasoning tier earns the cost.
Dev-E	Sonnet 4.6	$2	$15	Long implementation sessions, large context, polyglot. Sonnet is the right tier for the work.
Review-E	Opus 4.7	$1.50	$8	The acceptance gate runs the highest-reasoning model. Review quality is the place not to skimp.
iBuild-E	Opus 4.7	$3	$20	Substantial build sessions. Xcode, signing, App Store submission. Same reasoning tier as Review.

FOUR LAYERS OF COST CONTROL

Prompt cache. Anthropic prompt caching cuts input cost by up to 90% on repeated context. BRAIN.md and AGENTS.md are cached aggressively. The same long context, charged once.

Model routing. Match the model to the task. Haiku for classification, Sonnet for implementation, Opus for planning and review where reasoning quality compounds. The wrong model on the wrong task is a 10 to 30x overrun.

LiteLLM proxy. Per-agent virtual keys, per-key spend limits, automatic 429 fallback between providers. Currently planned, not deployed.

Kill-switch. Hard per-session cap. If an agent exceeds budget mid-task, the proxy returns a synthetic 402 and the agent escalates to #admin. Lives in LiteLLM, so honestly: not yet live.

09 HONEST GAPS

What does *not yet* work, named out loud.

A whitepaper that papers over its own limitations earns no trust from anyone who has shipped software in production. The pitch case for the rig is honest because the gaps are. These are the real things that are not yet right. The reason most of them are not right is that they are second-priority, not impossible.

WHAT SHIPS TODAY

- ✓ Issue dispatch with exactly-once delivery, on the Marten event store
- ✓ Dev-E in three runtimes (Node, Python, and .NET) with Python paused intentionally
- ✓ Review-E approving and blocking PRs on schedule, every PR, every day
- ✓ Memory MCP serving reads at session start with hybrid vector plus full-text search
- ✓ Branch protection and CODEOWNERS as the human veto layer
- ✓ Cost dashboard with per-agent attribution and live spend tracking
- ✓ Flux GitOps reconciliation for every rig component. The cluster is what the repo says
- ✓ Per-pod XREADGROUP partitioning so two consumers never claim the same work
- ✓ Dangerous-command guard at the shell-hook layer, refusing sudo, rm -rf, and friends before execution

WHAT IS MISSING OR BROKEN

- — LiteLLM proxy not deployed. Hard cost ceilings are advisory until this lands
- — Langfuse not deployed. Agent traces do not yet flow to any observability backend
- — External dashboard not exposed. Cluster-internal only; operator needs port-forward
- — Memory deployed but unexercised. The save pipeline is broken. Agents are not yet emitting the Learnings sections it scrapes for
- — Drift detection canary suite is designed and partially scaffolded; nothing alerts on it yet
- — Multi-agent epic orchestration is manual. Cross-repo work still threads through the operator
- — Memory poisoning is a known unsolved attack surface. A compromised Dev-E could write malicious memory Review-E later retrieves. Attested writes and tiered trust are designed, not yet deployed

THE HIT-USED METRIC IS FICTION, HONESTLY

The memory dashboard surfaces a "hit-used rate" intended to measure how often loaded memories actually influence agent output. Today it counts regex matches against a token format the agents do not emit. Observed hit rate: zero percent. The metric is disabled in the dashboard until citation-enforced retrieval or LLM-as-judge sampling replaces it. Documenting the gap is more useful than hiding it.

10 ROADMAP

Six months of *floor before features.*

Each item below is independently shippable and closes a specific gap named on the previous page. The order is dependency-correct: foundations before features, observability before optimisation, single-step automations before multi-agent orchestration. The roadmap is engineering, not vision.

Q3 2026

Raise the floor

LiteLLM proxy deployed, hard cost ceilings live

Langfuse self-hosted, agent traces flowing

Public conductor dashboard via Cloudflare tunnel

Memory save pipeline working end to end

Planner-E in production with regression tests

Drift detection canary suite alerting on baseline drift

Q4 2026

Multiply across projects

Onboarding runbook: zero to first agent-shipped PR in under one hour

Ops-E for deployment automation, agent plus human approval

Judge-E for LLM-as-judge quality reviews on a 5% sample

Multi-tenant memory scoping enforced at the query layer

Hard cost ceiling kill-switch validated against runaway scenarios

SWE-bench Pro evaluation harness running nightly

BEYOND 2026

What the rig becomes

Architect-E for system-level design proposals

Cross-org rig serving multiple companies' codebases

Property-based testing as the agent-output validation surface

Repair-E for SLO-gated automatic rollback

Drift-detection as a fifth channel covering memory itself

Productised onboarding for partner startups under the technical co-founder model

ONE SENTENCE TO TAKE AWAY

Five iOS apps. The web properties around them. The infrastructure underneath. The next of each is a config commit, not a hire. Adding operators multiplies the rate.

11 THE FOUNDER

Seventeen years of *building someone else's leverage.*

I have spent my career as the engineer companies hire when their platform is the thing that has to work. I am now applying that same discipline to my own company. The rig is what I built when I stopped accepting that engineering throughput had to scale with engineering headcount.

NAME

Stig-Johnny Støbakk

ROLE

Founder, Dashecorp AS

BASED

Oslo, Norway

EDUCATION

B.IT, Computer Science. Deakin University, Australia

LANGUAGES

Norwegian (native), English (fluent)

EARLIER IN LIFE

Fisherman and soldier before the IT degree. That shaped the work ethic more than the desk did.

2026	Dashecorp AS, founder. The company that holds the rig and the iOS portfolio. Six months of build behind the platform; the legal entity is new.
Invotek	Invotek AS, CEO. Currently embedded as DevOps and platform engineer at Norsk Helsenett, on the team that runs helsenorge.no.
2018 to 2022	Q-Free ASA, lead developer. Led distributed engineering across Norway and Portugal, building tolling and traffic infrastructure used at national scale.
2016 to 2018	Visma Consulting AS, lead consultant. Senior platform engineering across financial services and government clients.
2009 to 2016	Avento AS, senior consultant. Seven years building web platforms across .NET, EPiServer, and the cloud-native shift.
2009	Bachelor of IT, Deakin University. Three years in Australia, major in Computer Science and Software Development.

The short version: I have written and shipped production software for seventeen years across .NET, Node, Kubernetes, Azure, and GitOps. The rig is the platform I was building toward the whole time without knowing it.

12 THE ASK

What the next twelve months *fund.*

The Rig has been in production for six months, five iOS apps ship through it, and the next operator added to the platform multiplies output rather than headcount. Dashecorp is raising a seed round to close the named gaps, ship the partner-deployment thesis, and bring the first non-founder operator onto the rig.

THE THESIS · IN ONE PARAGRAPH

A traditional small engineering team is two to four senior engineers at roughly $200K each, all-in. The Rig replaces the routine half of that work, claims, codes, reviews, merges, deploys. One operator with the Rig matches the throughput of several engineers without it, and the per-operator ratio holds as the team grows. The bet for Dashecorp is to scale a portfolio of small, focused consumer apps on this leverage, and to deploy the same Rig for partner startups under a technical co-founder model. Internal use runs on subscription-tier billing; partner deployments run on metered API. The architecture is unchanged across both.

WHY NOW

The platform has been in production for six months and has stabilised on the architecture this whitepaper describes. The gaps on page thirteen are second-priority engineering work, not unsolved research. The next twelve months are not an engineering problem but a capital-formation, distribution, and operator-discipline problem. The next dollar of investment buys throughput and reach, not architecture.

WHAT I BRING

A working autonomous engineering platform shipping real software in production. Seventeen years of platform and distributed-systems engineering, including national-scale healthcare infrastructure at Norsk Helsenett. Founder-level ownership of every line of the stack, the Rig, the agents, the apps, the brand, the legal entity. A track record of leading distributed engineering teams across Norway and Portugal at Q-Free. Norwegian operating discipline applied to a category most operators are still treating as research.

WHAT THE NEXT TWELVE MONTHS LOOK LIKE

Close the gaps on page thirteen so the platform is institutional-grade by end of Q3, LiteLLM proxy live, Langfuse observability deployed, memory pipeline working end to end. Take Cuti-E from launching to a first paying cohort by end of Q4. Onboard the first partner startup to the Rig under the technical co-founder model, on metered billing, in Q4. Hire the first non-founder operator at Dashecorp around the same time.

WHAT THE ROUND FUNDS

Eighteen months of runway for the founder operator, the first non-founder operator, and the infrastructure budget to run the partner-deployment thesis without it cannibalising internal capacity. Honest conversations with investors who want to understand what is right and what is early. A cap table that gives Dashecorp the floor to operate at venture pace through 2027.

The portfolio ships. The rig that ships it *is the product.*

The rig is real. It ships iOS apps, web properties, and its own infrastructure under operator governance. The economics are real. The gaps are real and named. The next six months close the most consequential of them. This document is the snapshot of where the work stands today, and the platform expansion is what the next round funds directly.

THE FOUNDER

Stig-Johnny Støbakk

Oslo, Norway

post@stigjohnny.no

THE COMPANY

Dashecorp AS

dashecorp.com

nutrieapp.com

cuti-e.com

THE APPS

Nutri-E

Fast-E

Count-E

Reward-E

Cuti-E

WHITEPAPER · V1.5 · MAY 2026 · SET IN EB GARAMOND AND INTER · © DASHECORP AS