

STRATEGIC REPOSITION

**A new type of application
is being built everywhere.**

History.

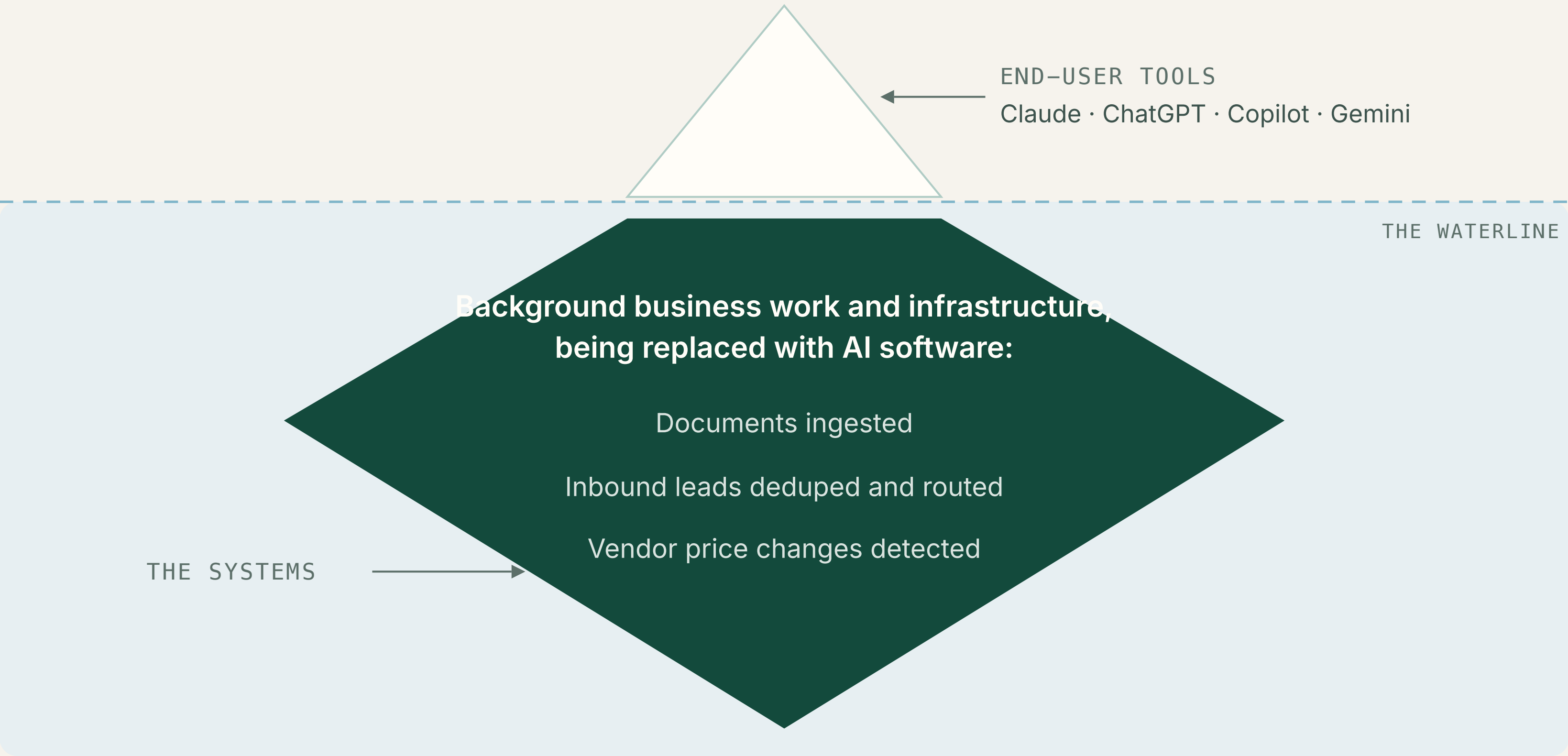
- 01 **Bet one.** Reliable AI comes from chaining bounded prompts into a deterministic process, not one large prompt. Now industry consensus.
- 02 **Bet two.** A visual canvas, so end users could construct their own solutions. We watched them use it and not build good ones: the work stayed trivial, and today an agent covers it anyway.
- 03 **Bet three.** Companies would need an internal portal to distribute AI applications: no major provider had an internal prompt library or supported external MCP. As models have commoditized, the mainline providers have focused on building lock-in through UX, PowerPoint in Claude, deep research in ChatGPT, and have added MCP support.

Reflecting on what succeeded.

There are two types of AI applications in a company:

- 01 **Tools** an end user builds, useful to one person at a time. They do not necessarily scale, and they are being absorbed in multiple places: qbash, Claude agents, ChatGPT agents, Cursor automations, Zapier.
 - 02 **Systems** that replace units of work in the business. These have stayed critical in almost every account, even as companies change chat providers, adopt other tools, and users evolve. These tasks run hundreds of times a month, may never engage a user at all, and keep running because operations depend on them.
-

The iceberg.



Replacing the work with Tasks.

We build these systems as Tasks: the process in deterministic Python, the judgment in bounded model calls. What building them this way buys:

- **Predictable spend.** Model calls are written in the code and loops have known bounds. A run is priced before it executes.
- **Your IP, abstracted from the model.** The process lives in your code. Swap the model without rebuilding the application.
- **Multi-model.** Cheap models for the bulk steps, a frontier model where judgment is hard, chosen per step.
- **API calls controlled by code.** Integrations are direct calls at lines you wrote, not tools a model chooses.
- **Privacy.** Data moves through your code and reaches a model only when you pass it.
- **The same steps every run.** Deterministic execution: code owns the flow, the model answers bounded questions inside it.
- **Versions.** Every save is a version with a line-level diff and one-click restore.
- **Tests.** Unit tests run in a sandbox, so a change is proven before it goes live.
- **Logs.** Every run records its inputs, output, errors, and duration.

Comparing the alternatives.

	Agent or Skill	Workflow canvas	Build it yourself	qbash Task
THE ARTIFACT	A prompt locked inside one provider. The process lives in the model's head, with no artifact to review.	A node graph in proprietary JSON. Models can't read it, and neither can a pull request.	Python on a VPS, plus the auth, deploys, monitoring, and model bindings you now own.	Python with the framework in scope. Diffable, reviewable, readable by the next developer or model.
COST PER RUN	Scales with how much the model thinks. The path varies per run, token prices rise over time, and a hard input can multiply spend with no ceiling.	Per-operation pricing that climbs with volume. Predictable until the logic outgrows the plan.	Cheap to run, expensive to own. The real bill is engineering time, not tokens.	Deterministic steps cost no tokens. Model calls are written in the code with known bounds, so a run is priced before it executes.
CHANGING IT	Rewrite the instructions and re-test by feel. No diff, and no way to prove behaviour before it is live.	Reconstruct the logic across nodes and re-wire by hand. Review means clicking through the graph.	A deploy pipeline and a DevOps backlog of your own. Model deprecations and SDK churn are your problem.	Line-level diff on every save, tests in a sandbox, one-click restore.

- triage.py · written as a Task

```
# the code you always could have written
issues = qbash.integrations.jira.search_issues(
    "status = Open AND team IS EMPTY",
)

for issue in issues:
    # the line you couldn't, until now:
    # which of 12 teams is this?
    triage = qbash.ai.run_prompt(
        slug="triage-issue", variables={"issue": issue},
    )

    qbash.integrations.jira.update_issue(
        issue["key"],
        team=triage["team"],
        priority=triage["priority"],
    )
```

- the same job, written for an agent

Go through our open Jira issues that have no team assigned. For each one, read it, decide which of our twelve teams should own it and how urgent it is, then update the issue. Keep going until everything is triaged.

Oh, and if an issue looks like a duplicate, link it to the original instead of assigning a team. If you cannot tell which team owns it, drop it in the triage backlog with a short note on why. Skip anything already assigned, do not touch closed issues, and if something looks like an outage, mark it urgent and post a heads-up in the incidents channel.

Tasks work everywhere.

A system built now has to survive an environment that will not hold still. Models are deprecated and replaced. Chat providers get swapped. New surfaces appear that did not exist when it was written. The deck opened on exactly this churn, and it is the permanent condition, not a phase.

Software has always answered a moving environment the same way: keep the logic in the middle fixed, and treat everything it connects to as an interface you can re-point. A Task is built that way. The model is an interface, the integrations are interfaces, the surface it runs from is an interface. The process is the only fixed thing, so a provider can change and the implementation does not.

Because of that, the same Task can be used flexibly anywhere:

Claude

ChatGPT

Microsoft Copilot

Gemini

Cursor

Slack

Gmail

Jira

Email

API

Schedule

04 · TODAY

Today.

- Programmatic Tasks now dwarf all other usage on the platform and are growing faster than anything we have shipped.
- They proliferate through an organisation: the prime builder lands, ships a system, and the rest of the org becomes users of it.
- SEO flywheels bring traffic on the specific terms a builder searches mid-problem.
- Agencies that build these solutions for clients are partnering with us: building and maintaining them on qbash is easier than on infrastructure they run themselves.

The thesis.

- Our thesis: these automations are replacing what would have been built as legacy applications, plus a large set that was never built because it was too expensive or too complex. As capabilities improve, they replace more of legacy application development.
- DigitalOcean grew **29%** year over year. Anthropic's run-rate revenue crossed **\$47B** in May 2026, and a large share of usage is repeated actions and repeated systems, not conversation. The growth gap is application creation moving to the model providers.
- Anthropic's September 2025 Economic Index found **77%** of business API usage was automation-patterned.
- Inside DigitalOcean, the same pattern: AI customer ARR grew **212%**, outpacing total revenue growth of **29%**. Raw materials, sold to people building applications elsewhere.
- The rest lands on canvases (Zapier, and n8n, valued at **\$5.2B** in May 2026) and in code assistants. Each one is an application that in 2019 was a dev-shop project on a VPS.
- The class exists at volume today, in chat threads, node graphs, and cron jobs. Very few are treating it as its own lane yet: the market reads it as an agentic runtime for end users.

The players, and their incentives.

The labs

OpenAI and Anthropic want to own the interface: app stores, agent modes, front-ends. A multi-model layer makes their models swappable, so building it runs against their incentive.

The clouds

AWS AgentCore, DigitalOcean Gradient, Azure AI Foundry: primitives priced per vCPU-hour and per token. Their incentive is selling more compute, not packaging it. Same gap Heroku exploited against EC2.

The canvases

Zapier and n8n are incentivised to protect the no-code artifact, and moving to code means abandoning their medium. We know, we had to.

Heroku

Moved to a sustaining engineering model in February 2026, emphasizing stability and support rather than new features. The closest named incumbent narrowed the ambition of its core platform.

The natural question: what happens when OpenAI and Anthropic absorb this, as they did the workflow builders? We are stepping into a level of application they are disincentivised to touch.

WHERE THIS GOES

We think this class of application is about to be everywhere. We intend to be the best place to build it.

The clouds sell silicon and tokens. We're the application layer the tokens are for.