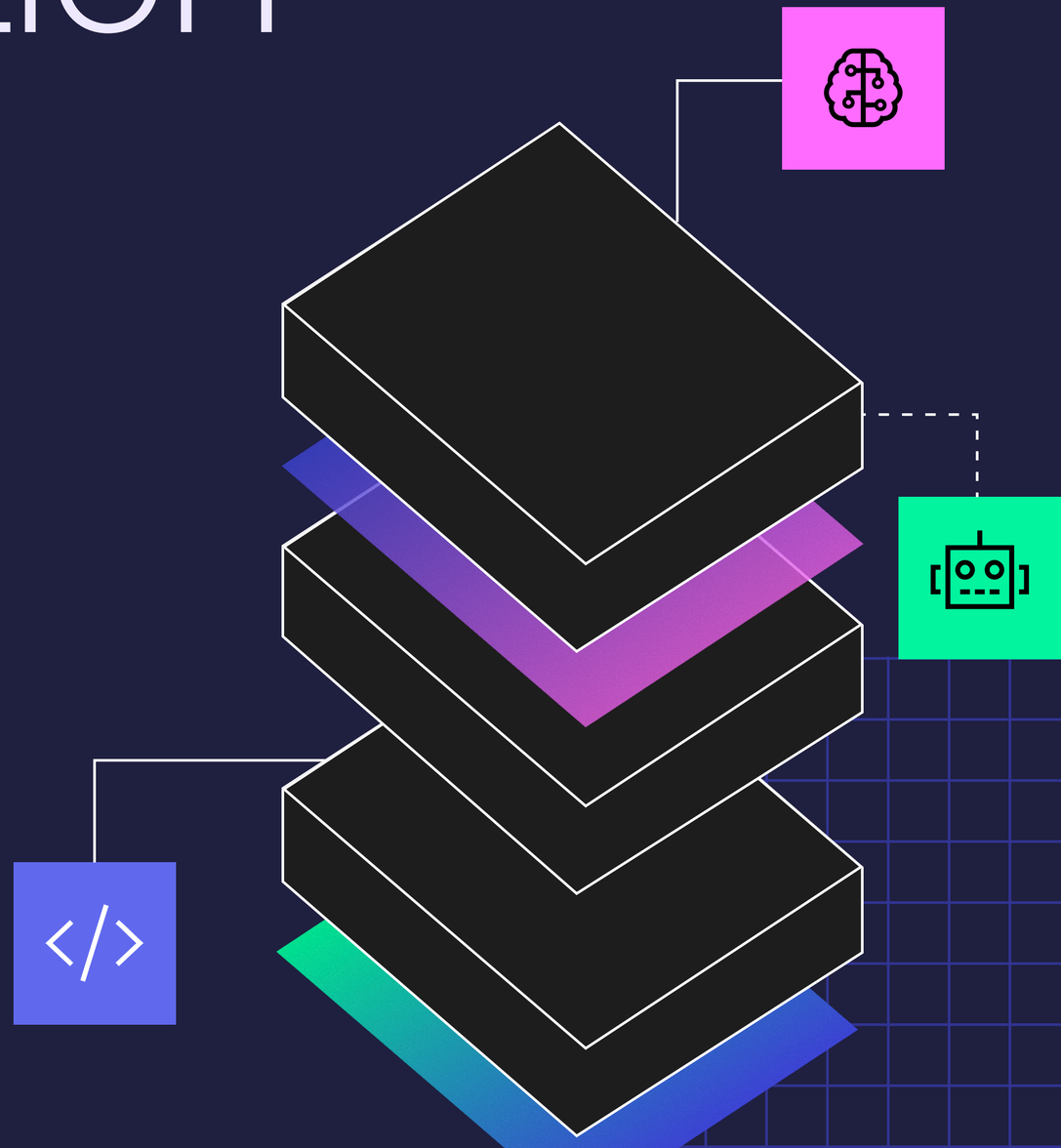
The AI production stack report

An analysis of AI complexity, risk, and Durable Execution in 2025

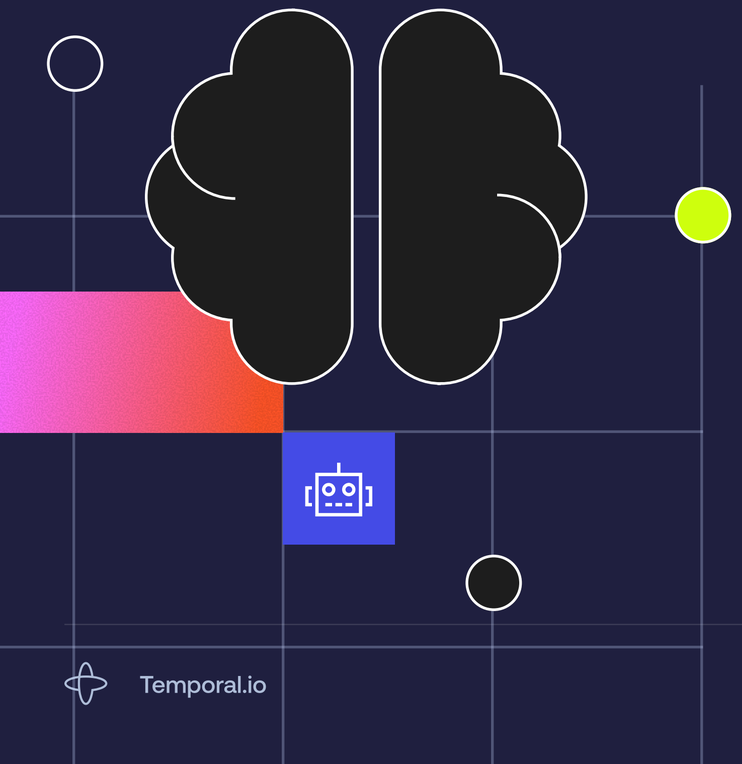


Temporal

Table of contents

INTRODUCTION

AI has a brain, but no backbone



Spinning up an AI agent has never been easier. With a goal in mind, a few minutes of your fingers flying across the keyboard, and a handful of APIs, you'll have something that will make your fellow devs' eyes sparkle in a demo.

The bigger question is: will it provide value for your users?

What looks good in the demo often masks much deeper issues.

The reliability? Shaky.
The observability? :shrug_emoji:
The state management? Messy.
The development complexity? Quite overwhelming.

But, we're curious. And we're always eager to learn more about the software industry and the developers who fuel it, so we conducted research with the goal of understanding how today's engineering teams navigate AI challenges.

To achieve this goal, we surveyed more than 150 developers and technical leaders across a wide range of industries. We dug into the tools, languages, and infrastructure behind all their AI projects and figured out where the cracks show up and, as a result, what a production-ready AI stack needs to survive.

The story is honestly pretty straightforward: **today's AI tooling makes it easy to build prototypes, but nearly impossible to build production systems with actual value.**

This report serves as the guide to help you bridge the gap between an AI prototype and a system that delivers business value.

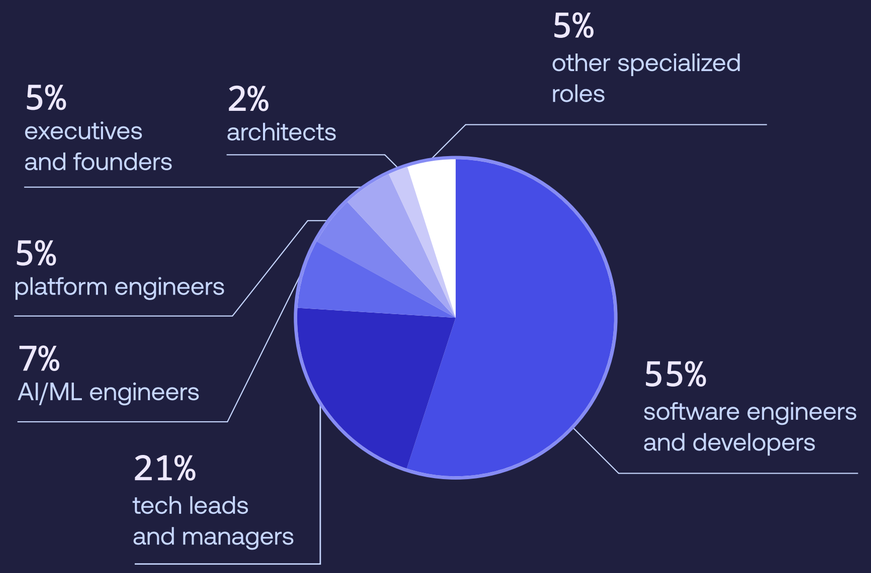
Who we heard from

So, who exactly did we ask to open up to us about their AI headaches?

153 diverse respondents

Roles

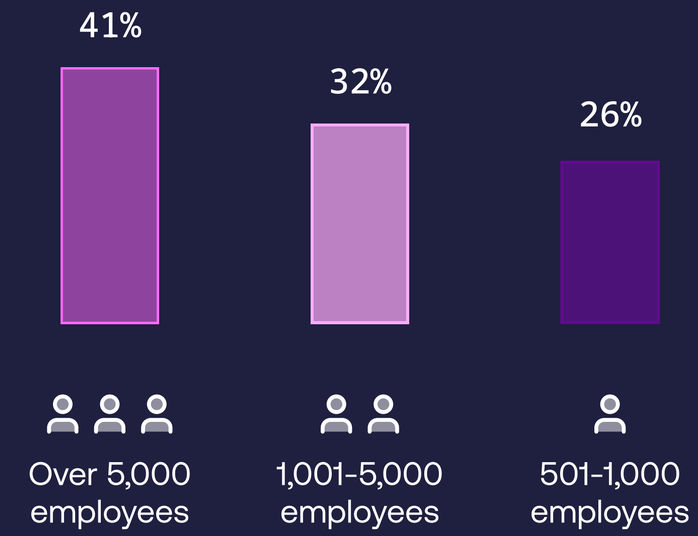
We made certain to select respondents who represent a wide range of people. More than half **(55%)** are software engineers and developers actually shipping code. Another fifth **(21%)** are tech leads and managers trying to wrangle those engineers (and their systems). The rest are, AI/ML engineers **(7%)**, platform engineers **(5%)**, executives and founders **(5%)**, and architects **(2%)**, along with a small set of other specialized roles.



Company size

Company size tells its own story: **41% come from enterprises with over 5,000 employees, another 32% from companies with 1,001–5,000 employees, and 26% from organizations with 501–1,000 employees.**

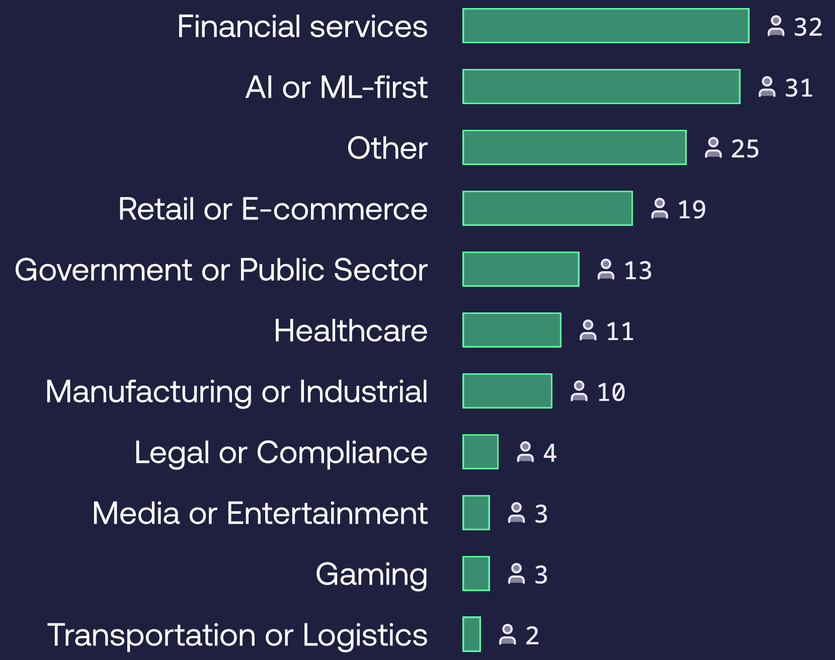
Everyone here is probably someone just like you: a member of a large company trying to run AI at industrial scale.



Who we heard from

Industry

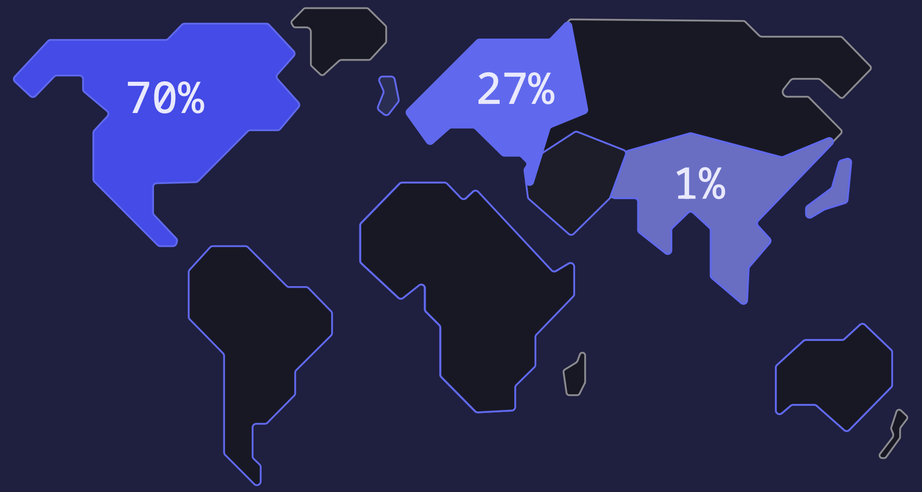
Industry spread is broad, but a few stand out: Financial services **(21%)**, AI- or ML-first companies **(19%)**, retail and e-commerce **(12%)**, government/public sector **(8%)**, healthcare **(7%)**, and manufacturing/industrial **(7%)**.



Region

And finally, we took our research worldwide. 70% of our respondents come from North America, 27% from Europe, and 1% from Asia-Pacific.

Different regions, but surprisingly, the same struggles. Whether coding in California, debugging in London, or firefighting in Bangalore, the reliability gap is global.



North America 70% • Europe 27% • Asia-Pacific (APAC) 1%

Key takeaways

Need a quick rundown of the best info from the report? Here's what you need to know.

Curious but tentative

Nearly half are still early along in their AI journey: 30% said they're exploring use cases, and another 19% are prototyping in the development and testing phase. This means that **49% of people are experimenting more than executing.**

On the other end, **25% said AI is already essential to their business**, and **14% are actively scaling production systems.** Together, that's 38% in what we'd call the "mature" camp; already past experiments and moving into operational AI. The remaining **13% are piloting in production**, caught somewhere between those poles.

Confidence is key

Confidence in production tells a similar story. **Only 13% of respondents feel very confident** in their ability to observe and debug AI workflows at scale. Another **28% are somewhat confident.** But the majority fall into uncertainty or struggle: 38% are neutral, 15% not very confident, and **7% aren't confident *at all*.**

In short, almost **60% of teams lack the confidence they need** to keep AI reliable in production. Without better observability and evals to measure LLM behavior at scale, debugging is guesswork.

Money, money, money

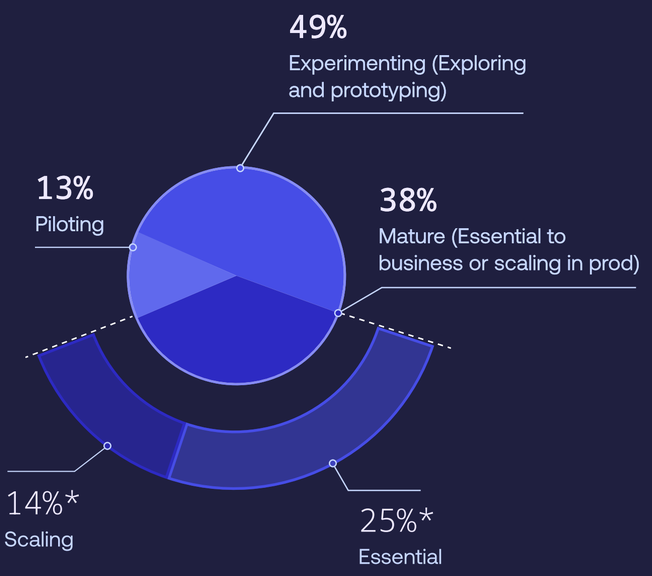
And the costs make a huge dent in the annual budget. **62% of teams lose measurable time or revenue** to reliability issues. For many, that's 10–50 developer hours / $1–10k in losses each year; for some, it's 200+ hours or $100k+ annually. Only a mere 10% said they've lost nothing.

What to remember: while experimentation is still the norm, production AI is already a reality for a minority, and reliability gaps are extracting a measurable tax on everyone.

As one of our respondents, a software developer at a 1,001+ employee company, said:

> "Some of the biggest issues involved when developing AI [are] data quality and quantity, scalability and computing power, [alongside] software reliability and malfunction."

AI adoption at a glance



*Exact percentages are 13.9% for scaling and 24.5% for essential

LLMs

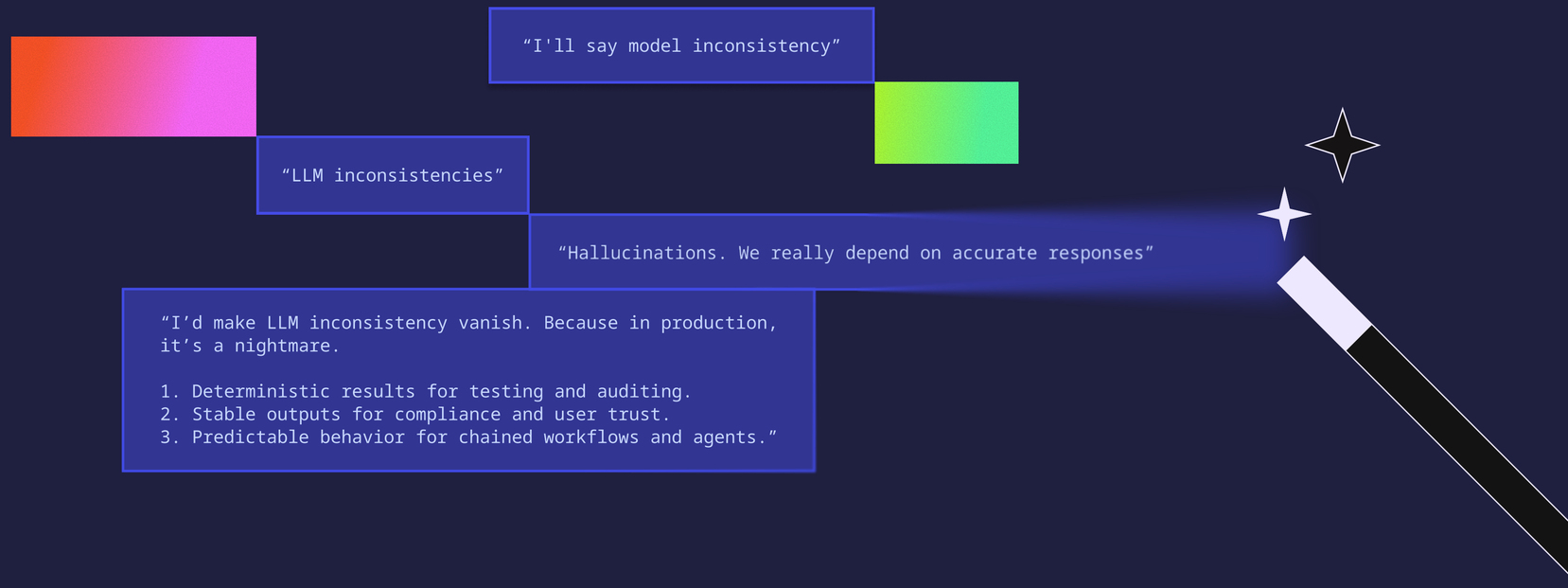
Now, let's drill down into your AI stack. This is a layered system rather than just a singular tool, right? Brains alone can't make a body. To function at an enterprise scale, you need layers that interlock: orchestration, databases, memory, LLMs, observability, and more.

Our survey showed just how fragile today's stacks are. A complete, solid stack should eliminate those struggles. Here's what it takes:

LLMS

Large language models (LLMs) are the reasoning core of modern AI systems. It's the place where summaries are generated, content is created, and decisions are made. But they don't run in isolation. **61%** of our respondents told us that **LLMs are already part of their current AI tech stack**, however, a whopping **48% also said that LLM inconsistency is the most fragile (or difficult to scale)** part of their AI system.

Respondents repeatedly noted that LLMs introduce brittleness through means of timeouts, throttling, and inconsistent outputs that break downstream systems. In fact, hallucinations and inconsistency were a top pain point often mentioned when asked what one issue respondents could "wave a magic wand" and get rid of:



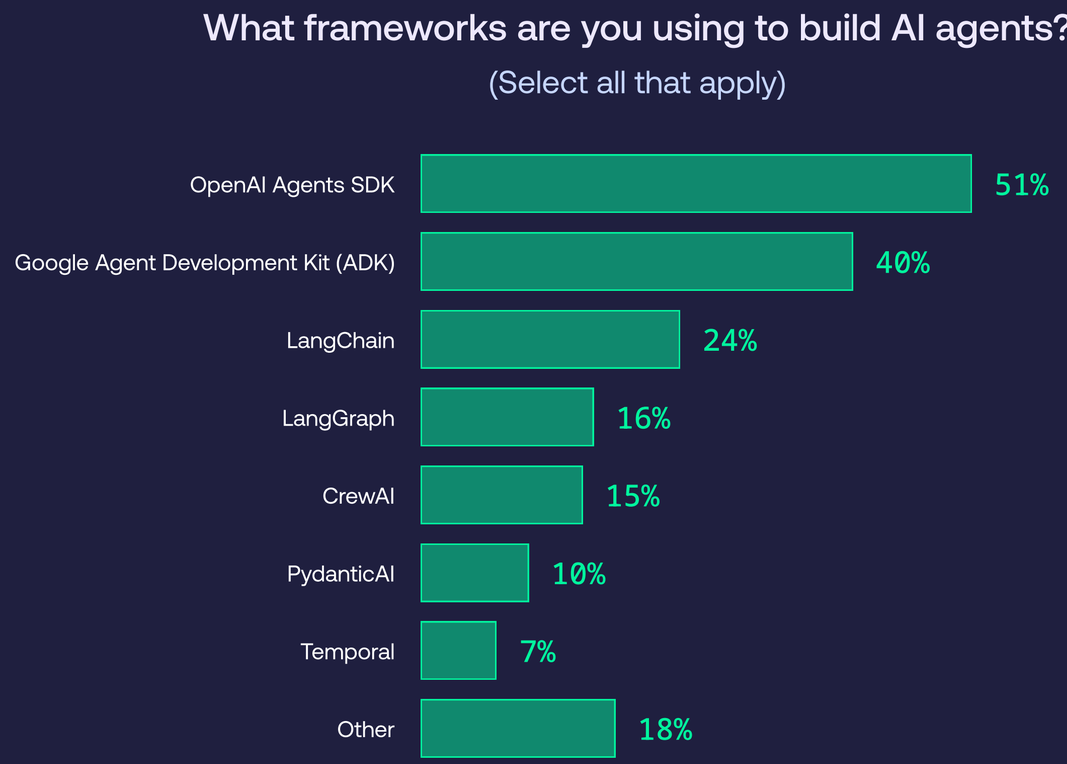
Agent frameworks

Now, onto agent frameworks. Agent frameworks give structure to LLMs and do so by taking a goal and breaking it into steps.

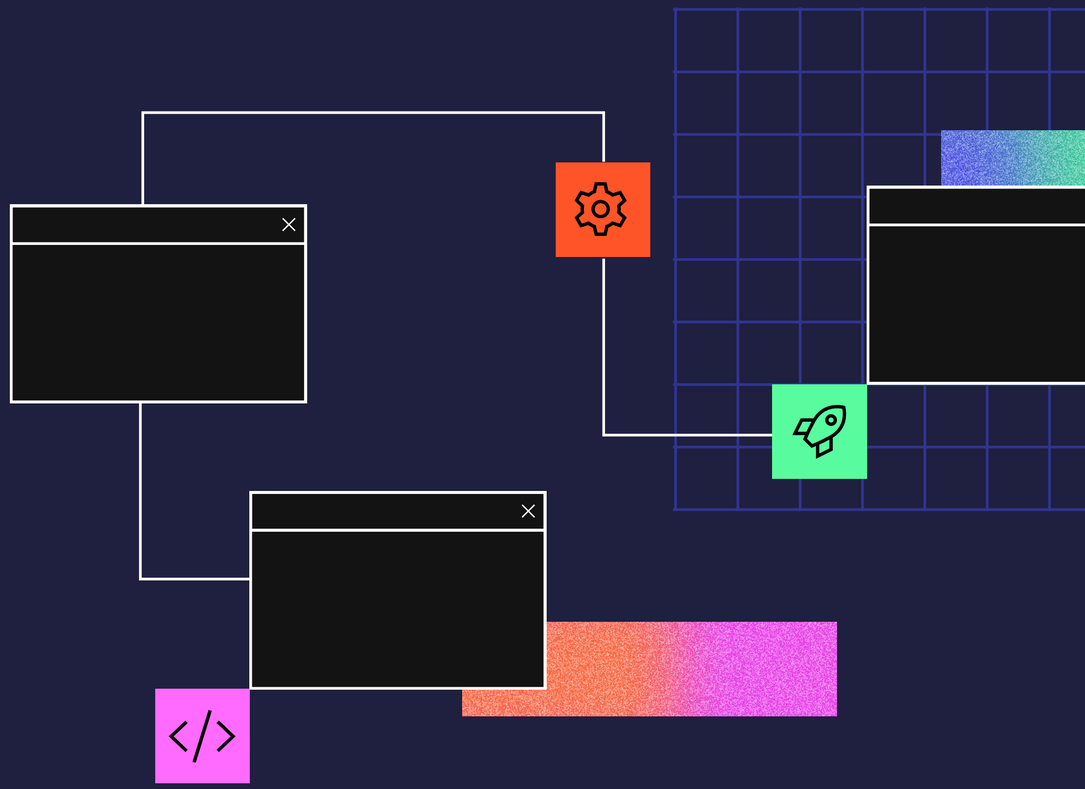
Wish you had a crystal ball to tell you what your competitors are doing? Look no further than our data.

51% of respondents told us that they're using **OpenAI Agents SDK** to build their AI agents followed by **40% using ADK (Google Agent Development Kit)**, and then **LangChain at 24%**.

Given that OpenAI Agents SDK tops the list, it's worth noting: Temporal now integrates directly with OpenAI's SDK. It makes it as easy as possible for your team to bring reliability and orchestration into the agent frameworks you're already adopting.



Agent frameworks



Frameworks may enable fast iteration, but they rarely carry teams across the gap to production reliability.

The frustrations came through clearly in comments from our research: frameworks are stateless by default, they lose context across sessions, and when they fail, they do it in the background.

One respondent, a Senior Software Developer at a company with 1,001+ employees, summed it up nicely:

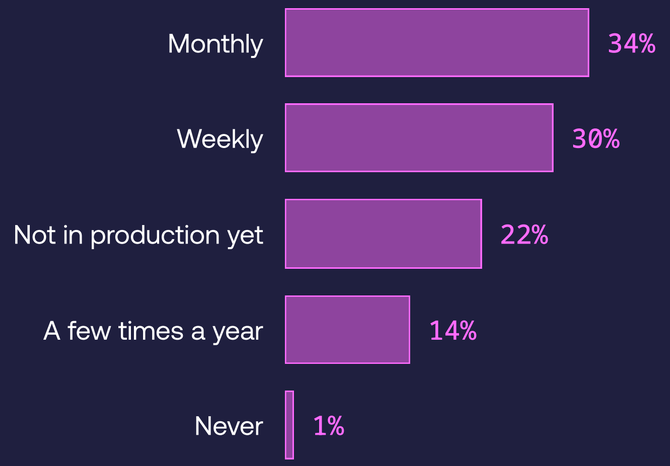
> “Frameworks are not very mature. Documentation is not up to date and integrations have gaps (I am thinking mostly of [recently released tools], but this applies to all frameworks we have been using). Challenges with LLM memory systems.”

Frameworks on their own don’t provide orchestration or observability, so they need to be paired with an external system; one that can track state, handle retries, and give you visibility into what’s happening. Without this, you’re only adding structure to an ongoing fragility issue.

Memory

Beyond the frameworks, memory is what gives AI systems continuity. Without it, every interaction starts from zero, and your "smart" system ends up with the recall of a goldfish *glub glub.*

How often do you encounter issues in your AI workflows?

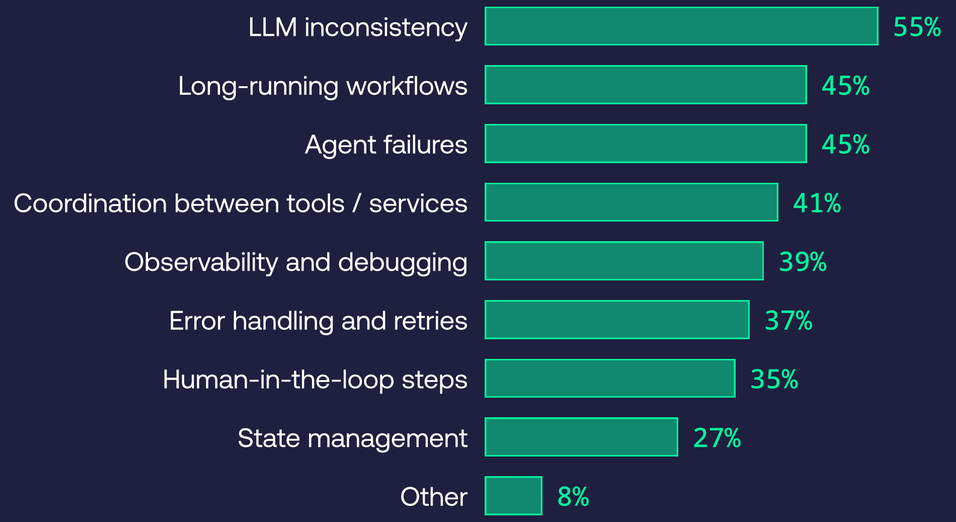


Respondents were blunt when telling us that memory is among the most fragile layers in today's AI stacks. **In fact, a third of respondents told us they encounter issues with AI workflows *on a weekly basis.*** This is a level of failure that no amount of patchwork can sustainably cover.

We dug even deeper into the minutiae of these AI systems to learn more.

Which aspects of your AI systems are most fragile or difficult to scale?

(Select all that apply)



Memory

Among the respondents who identified a memory component as part of their AI stack, fragility still showed up in multiple layers. While **27%** cited their state management as being fragile, even more flagged upstream issues like **LLM inconsistency (55%), long-running workflows (45%)**, and **agent failures (45%)**. All the top places where state is most likely to break.

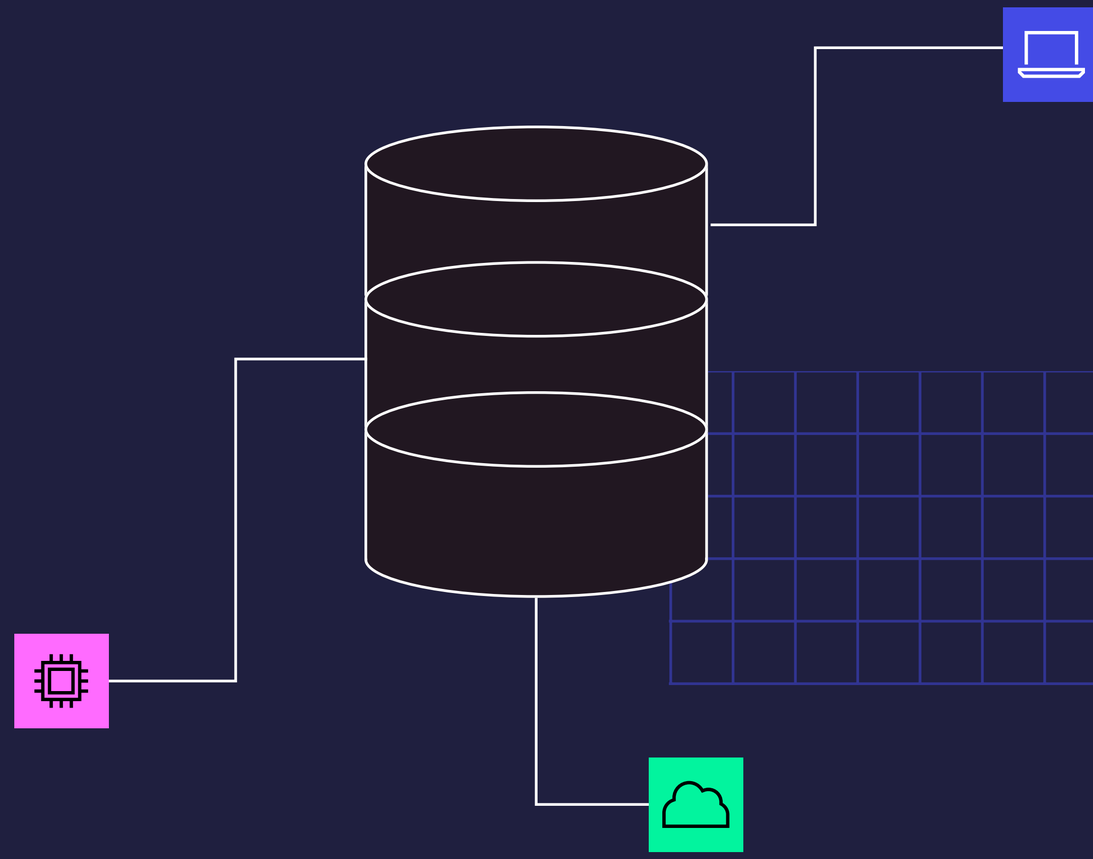
In other words, yes, memory may help you recall the past, but it doesn't guarantee reliable state in the present. Without dependable orchestration, agents go off the rails and developers end up debugging state by hand.

Another Senior Software Developer, this time at a company with 5,000+ employees, described the pain of developing AI applications this way:

> "Testing locally can be challenging due to model requirements for memory / GPU processing. A lot of development relies on "print debugging" / examining remote traces / logs."

Rather than waiting for your dev team to be paged at 2 a.m., get prepared.

A production-ready stack treats memory like a scarce resource: carefully selected, filtered and managed. It curates the most important context, compacts history where possible, and offloads unneeded data for just-in-time retrieval. Without this, your memory becomes a junkyard of irrelevant details, leaving your agent helpless and stumbling over where to go next.



Databases

Moving on to databasesl: **78% of our respondents told us that databases are a current part of their AI tech stack,** but many still report dissatisfaction despite their prevalence.

Databases in the AI Stack, by industry

Which components are part of your current AI tech stack?

% of respondents using databases

Across industries, the adoption trend peaks in **manufacturing (78%)** and **healthcare (73%)**, where structured data is critical for uptime and compliance. **AI-first companies (63%)** also lean heavily on them, while **government and public sector lags at just 31%.**

Size matters too. Smaller companies **(51–200 employees)** report **77% adoption**, but that figure drops to **47% among the largest enterprises (1,001+)**. This shows that companies aren't struggling because they don't have enough databases, instead, they're struggling because they have far too many.

When data lives across multiple systems, regions, and formats, then orchestration becomes a big challenge. The sprawl and coordination overhead is what makes it harder to rely on a single database layer as that "backbone" of your AI system.

The frustration came through in the survey responses.

> One software developer from a 1,001–5,000 sized company was to the point when telling us the ***one thing*** that would best improve the reliability and performance of their AI tech stack is, "better databases for gen AI."
>
> We hear them loud and clear.

Now, while the ask is super simple, the gap is vast and structural. Developers want data that ties seamlessly into workflow state, so every query and every row can be trusted in production.

Bridging the gap between "stored" and "used effectively" requires orchestration that binds database operations to workflow execution. Otherwise, you're just integrating problems even further into the stack.

Tools

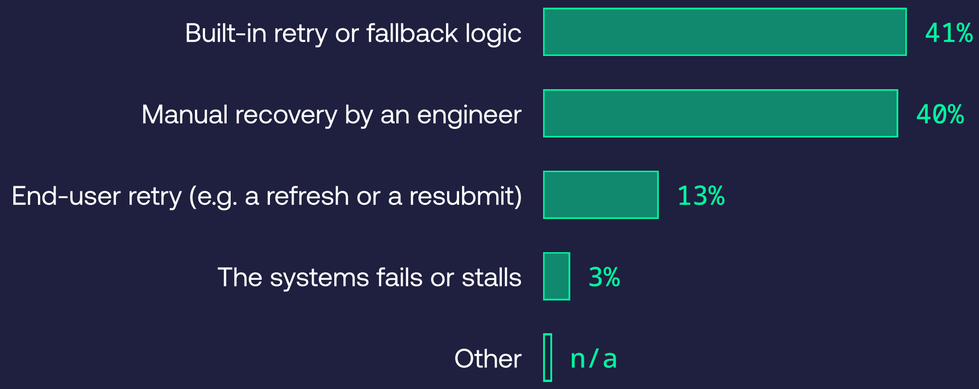
Every AI system depends on tooling to actually perform the services you need. APIs, services, and integrations all matter, but without orchestration, teams told us they're stuck rewriting the same error-handling logic over and over. Failures cascade when a single API call can't be retried mid-action.

The biggest signal in our data is confidence or, unfortunately, the lack of it in the industry.

How confident are you in your ability to use your current tech stack to observe, debug, and build production-level AI workflows?

What's literally presented here is an abundance of neutrality, but in the grand scheme of things, it's actually quite a bit of uncertainty. And uncertainty is a tooling problem: if you can't see and control your software, you can't trust your outcome.

When issues occur, how do you typically resolve them?



You should be able to trust your tooling to catch the majority of failures automatically (with edge cases requiring manual rescue). The problem is, too many teams face the opposite reality.

Manual recovery by an engineer is almost as common as built-in retry or fallback logic.
If you ask us, that's a failure of orchestration support.

Orchestration

Orchestration is the connective tissue between everything in your AI stack. If we're keeping up with our body analogy, we'd consider it the backbone that ties everything together. Without it, long-running, multi-step agent workflows fall apart.

The longer the agent runs, the more fragile it becomes. Basically, the more steps = the more chances for failure, more context to track, and more state to manage. Orchestration keeps it all intact.

Developers understand that it's important, with **47% noting that workflow orchestration is part of their tech stack** and **31% noting that they've gone as far as to build their own workflow orchestration in-house.**

Our data shows why this matters and where things still get sticky with **62% of respondents reporting measurable time or revenue lost due to reliability issues.**

Many of our respondents were very straightforward about the stakes:

- "Version control, orchestration, and tracing are all big issues when developing AI applications."

  — Technical Lead, 1–10 employee company

- "Orchestration failures."

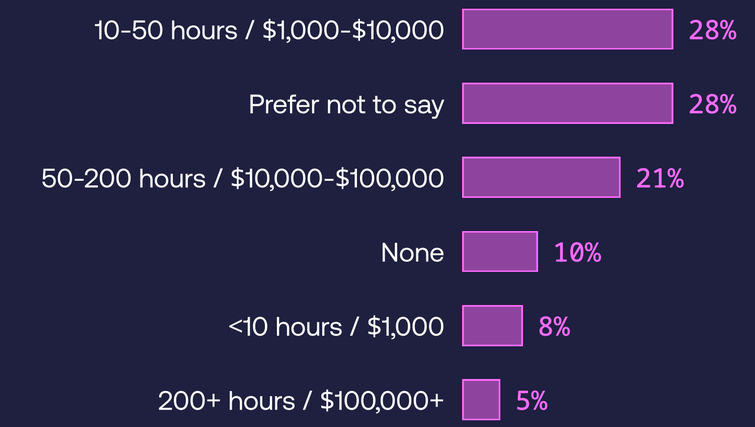
  — Software Development Manager, 5,000+ employee company (when asked what they'd eliminate with a magic wand)

- "The single most impactful thing to improve the reliability and performance of an AI tech stack is to build a robust MLOps framework that automates the entire lifecycle, from data management to continuous monitoring and retraining of models."

  — Software Developer, 501–1,000 employee company

It's easy to see the thread that runs through all three responses: teams can't keep fixing retries and recovery logic. In their words, version control needs to be something bigger than an afterthought, orchestration needs to work reliably, and automation that supports the full lifecycle of AI. And those are exactly the places where orchestration plays a vital role.

Roughly how much time or revenue do you estimate that your team has lost due to AI reliability issues?



Observability

Observability is where we found the confidence gap is most obvious. **42% of respondents** note that they're using an observability, logging, or tracing tool, **34%** find it so important that they've **built their orchestration tooling in-house,** and yet, **a third (31%) of respondents** cite "observability and debugging" as one of the most fragile or difficult to scale parts of their system.

Additionally, of the people who saw LLMs mess up (**48%** of our total respondent pool), about **a third (32%)** also noticed problems with a marked lack of observability. The fact that these observability and debugging issues overlap with LLM problems proves the larger point that LLMs are fragile (and teams can't figure out just how fragile they are as applications grow).

The outcome reflects terribly on the confidence levels in the industry. Only a meager **13% of teams** feel very confident debugging production AI, **while nearly 60% are neutral or not confident at all**. That's effectively a black box for the majority of teams.

This is where evals take on a vital role. While conventional metrics, logs and traces tell you if your system is running. **Evals** tell you if it's running **well**. While logs show, for example, that your customer service bot responded in 1.2 seconds with information, evals catch that the underlying LLM hallucinated a link to a support page that doesn't exist. Evals **systematically test model outputs** against ground truth (correctness), your actual data sources (hallucination), and even behavioral rules (tone, safety) to validate quality at scale.

Complete AI observability requires both layers: logs, metrics, and traces for system health, and evals for model correctness. One without the other leaves critical blind spots.

The tool of observability issues:

As one respondent told us:

"To improve the reliability and performance of my AI tech stack, **I would say implementing solid observability and monitoring is key**, it would help me catch issues before they blow up and keep everything running smoothly. With that in place, I could focus on fine-tuning the system and making sure it's always on point."

— Senior Software Developer at a 1,001-5,000 employee company.

Infrastructure

Infrastructure is the layer that turns prototypes into production systems. For larger companies, our data shows that this is where fragility grows fastest. Respondents at **enterprises with more than 1,000 employees were 2.5x more likely to report low confidence debugging production AI compared to smaller companies (25% vs. 10%).**

When asked “What are the biggest issues you see when developing AI applications?”, infrastructure came through loud and clear.

- “I have issue[s] with data quality and integration…”
  — Technical Lead/Manager, 201–1,000 employees
- “The biggest issues are poor data quality, bias, lack of explainability, scaling challenges, and ensuring the model stays accurate and relevant over time.”
  — Software Engineer, 201–1,000 employees
- “The biggest issue is integrating tools into the current infrastructure along with data privacy concerns.”
  — Software Engineer, 1,001 employees
- “My biggest issues are integration problems and high development costs.”
  — Software Engineer, 51–200 employees
- “High compute costs, security risks and unclear regulations.”
  — Software Engineer, 1,001 employees
- “Storage issues, latency issues and compute needs.”
  — Software Engineer, 1,001 employees
- “Having the infrastructure and place and knowing how/when to use it.”
  — Technical Lead/Manager, 1,001 employees
- “Data quality and availability and Integration with existing systems.”
  — Software Engineer, 1,001 employees
- “Integrating AI into our existing infrastructure.”
  — Technical Lead/Manager, 1,001 employees
- “The complexity of developing native AI applications without third party integrations, but third party integrations simplifies the processes better.”
  — Software Engineer, 1,001 employees

The quotes vary, but the sentiment is very consistent: infrastructure fragility is now more about cost, complexity, and fit more than it’s about a lack of raw compute power. Teams struggle to make AI workloads scale predictably, but even more, they struggle to make them integrate cleanly with systems that already exist (this is where evolution often gets hemmed up in bureaucracy) That integration gap is where costs spike, timelines stretch, and projects stall.

How your peers work with AI

The reality is no single platform is going to solve every problem in building production grade AI. In fact, we're sure that the teams we spoke with still wrestle with issues like wrangling data and evaluating model performance; but when it comes to orchestration and reliability (the problems that plague agents and workflows most) Temporal emerges as a great option.

Let's see this in action.

Replit: Developer tools at a global level

Background: Replit is one of the world's largest online Integrated Development Environments (IDEs), with over 20 million developers coding, collaborating, and deploying in the browser. Their leading product, Replit Agent, helps users scaffold, refactor, and extend projects with AI.

Challenge: Agent adoption surged, but reliability issues (memory crashes, provider outages, failed tool calls) and orchestration complexity (ensuring one agent process per user, multiplayer coding, infra spin-up) threatened the user experience.

Solution: Replit migrated the control plane to Temporal Cloud. Now, every agent runs as its own Workflow, with failure-prone steps handled as Activities that auto-recover.

Results: Migration time decreased from months to mere weeks because Temporal eliminated edge-case failures, supported multiplayer features, and scaled without becoming a bottleneck.

> It's a pretty bad user experience to have the agent get super far into something and then hit a catastrophic error, and you lose everything and have to restart.
>
> — Connor Brewster, Lead Engineer

ZoomInfo: Sales and marketing intelligence

Background: ZoomInfo is a market leader in B2B data intelligence, relied on by many sales and marketing teams to target and engage prospects in real time. Precision and timing are the bread and butter of the business.

Challenge: Their systems process millions of buyer signals daily, but brittle pipelines and batch delays made real-time personalization impossible. Opportunities slipped through the cracks when workflows couldn't keep up.

Solution: ZoomInfo turned to Temporal Cloud to orchestrate their workflows reliably. With Cloud, they were able to manage retries, state, and event-driven triggers at scale.

Results: Lag all but disappeared going down from hours to seconds, enabling precise marketing campaigns. Instead of losing opportunities to pipeline failures, ZoomInfo can now deliver personalization for their crucial use cases.

> "We've been able to scale up, and Temporal has never been the bottleneck."
>
> — Frank Show, Senior Principal Software Engineer

How your peers work with AI

Gorgias: AI agents for customer service

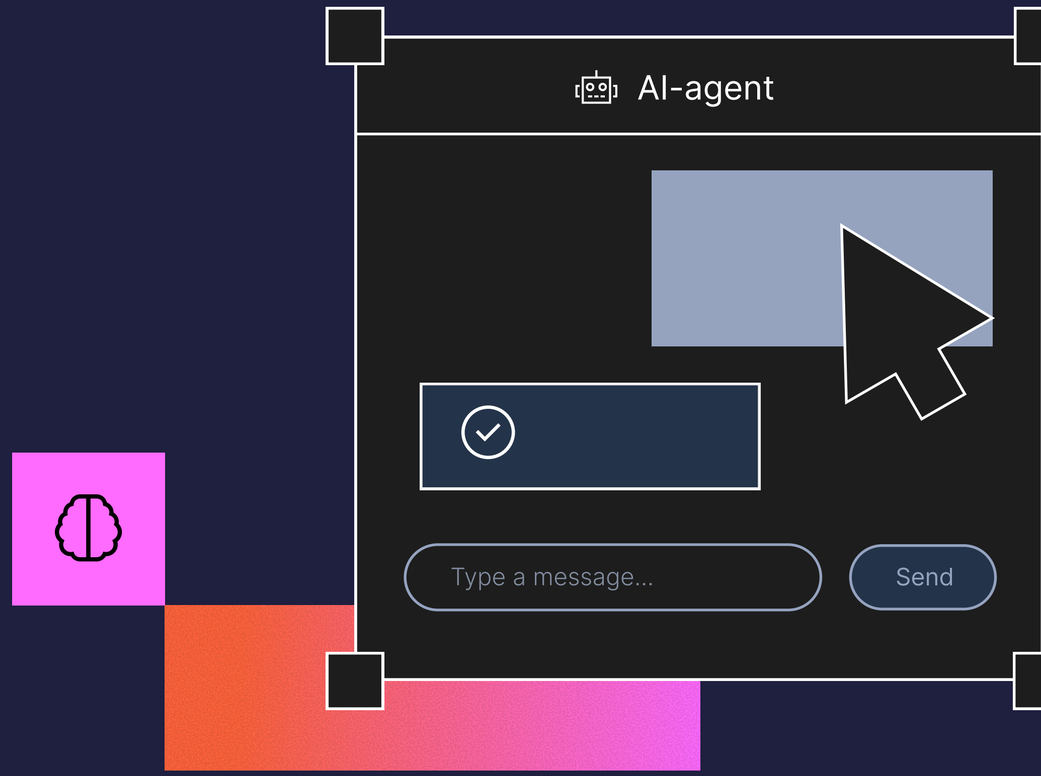
Background: Gorgias is an AI-powered helpdesk platform for e-commerce. Their platform serves thousands of merchants with customer service automation. They specialize in AI agents that can resolve complex tickets end-to-end.

Challenge: Conversations in a helpdesk platform are just as you'd expect them to be: long-lived, stateful, and fragile. Also prone to failures when LLMs hallucinate, APIs timeout, or sessions lose context, meaning that agents couldn't be trusted to finish what they started.

Solution: The team turned to Temporal Cloud and now every AI-driven conversation runs as a Temporal Workflow. Temporal keeps state, retries failed steps, and ensures agents don't "forget" midstream.

Results: Agents now handle multi-step, multi-system support issues without the need for a dev to dote over the process. Reliability freed Gorgias engineers from writing weak code and let them focus on agent quality and customers see faster resolutions with fewer dead ends; a win-win for everyone.

> "We're able to work with the agent again and again. It was really painful to do this without a workflow approach. I think for us, a workflow approach with Temporal was good because in the end, all LLM use cases are workflows."
>
> — Romain Niveau, Senior Engineering Manager



Where orchestration fits

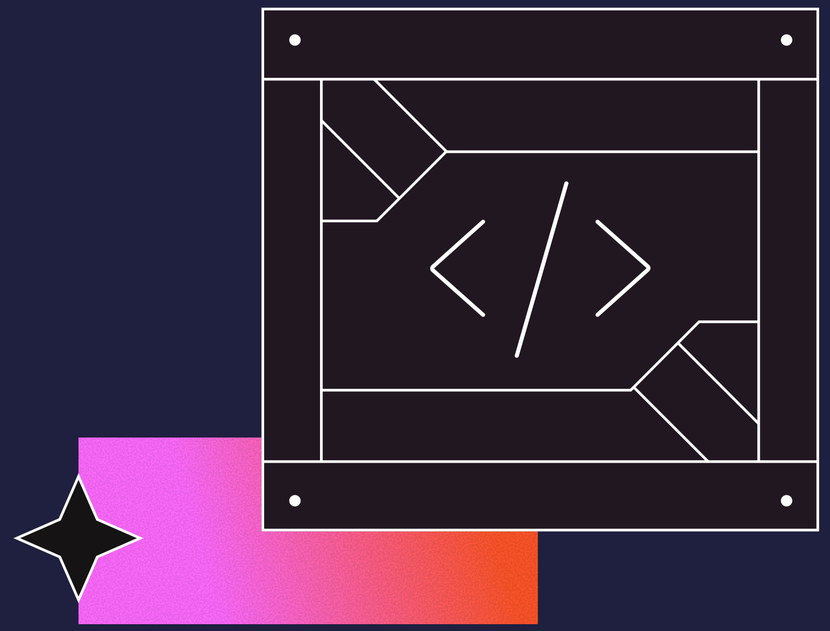
While we took the time to author this report, the data did all the heavy-lifting in the storytelling. It’s clear to us that today's fragility goes far beyond model issues. The silent killer of many orgs is weak orchestration.

Because 62% of teams told us they’re already losing measurable time or money to reliability issues and nearly 60% lack confidence debugging production AI, we know that these are systemic issues that need a powerful framework for resolution. This means that impeccable orchestration is table stakes for production grade AI.

Enter **Durable Execution**.

Durable Execution is the ability for workflows and agents to run reliably over time; automatically tracking state, retrying failures, and recovering from crashes without a developer needing to stand by babysitting.

In other words: the backbone today’s agent stacks are missing.



Looking toward the future

We know that the promise of AI today is real, and teams are clamoring to mobilize around it (particularly us devs tasked with making applications match the hype). It's easy to be consumed by the challenges of the present, but just as vital to remember that as AI becomes more of a mainstay, the expectations of your users will only continue to grow. You're busy now... but what about the future? Do you have a plan for the challenges down the road?

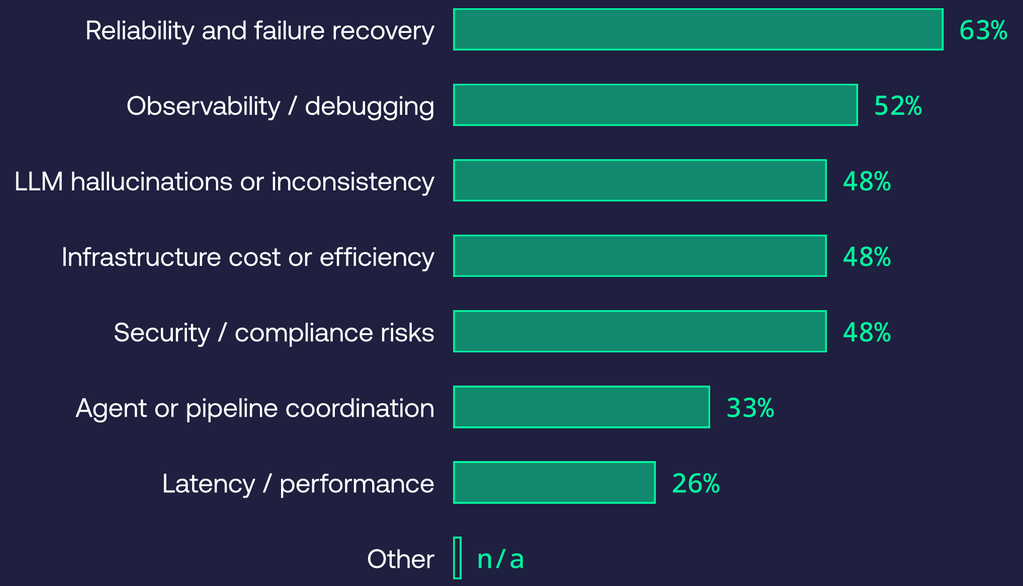
Many of your peers are already thinking about the future. When asked how much impact AI will have on their business in the next 12–24 months, 88% of respondents predicted at least a moderate boost to efficiency or revenue; with more than half expecting significant, or even transformative change.

How much impact do you expect AI to have on your future time/revenue? (12-24 months)

Growth won't come for free and teams know this. They're clear-eyed about the obstacles down the line because the top priorities for the next two years line up almost perfectly with the weaknesses they flagged earlier:

The takeaway is that AI is surely buzzing and the hype lies around smarter models, but the builders behind this know they need reliability, observability, and reliable infrastructure. Make sure your team is aware, too.

Which of the following things are most crucial to address in AI production in the next 12-24 months?



Building production ready AI that lasts

While it's easy to solely focus on your AI doesn't that fails due to bad model outputs, it's still important to pay close attention to the systems surrounding it that aren't built to endure — these have a profound impact too. Fragility shows up in the worst spots imaginable, like the retries that never fire and the workflows that lose state.

What builders told us is that another shiny framework will only add to their issues down the road. What they need is reliability and, while it might not be the flashiest requirement, it's what creates a product you and your customers can trust.

Give your AI a backbone.

Bring Durable Execution to your prod environment with a free trial of Temporal Cloud and $1,000 in free credits.

Temporal