IIElevenLabs

WHITE PAPER · 2026

Beyond the build: The operating model behind successful enterprise agent deployments

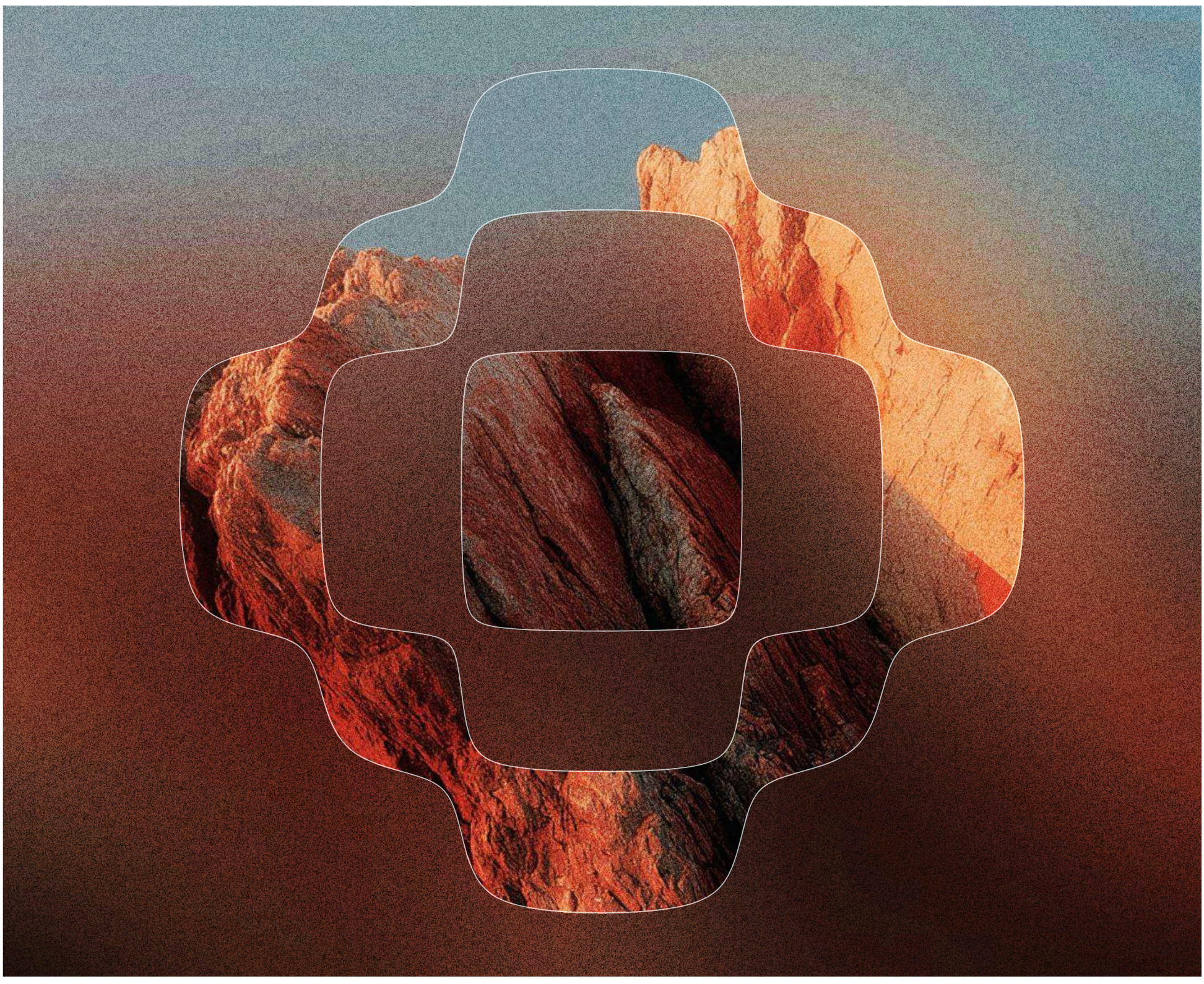
A field guide to bringing AI voice and chat agents to production - who owns them, how success is defined, and how they keep improving at scale.



Eli Goodman
Agents Growth · ElevenLabs



Caleb Wright
Forward Deployed Engineering · ElevenLabs



Agent projects typically don't stall because of technology. They stall because of the operating model around it - initial use cases that are too ambitious, success metrics that are not clearly defined, or relevant stakeholders who are not brought in early enough.

Deloitte finds close to three-quarters of enterprises plan to deploy agentic AI within two years, yet only one in five report having a mature governance model[1]. The deployments that succeed take a measured approach: starting with use cases tied to a business priority but manageable in scope, and building governance both into the agent and around it.

Creating a robust operating model for your agents will help you answer questions like:

- **Who owns the knowledge base when a company policy changes?**
- **How do you decide the agent is ready for more traffic?**
- **How do you know quality has dropped before customers tell you?**
- **What's the rollback plan if an update makes things worse in production?**

While the shape of every rollout varies, this piece reflects what we have seen work consistently and draws on real deployments powered by ElevenAgents.

Executive snapshot

KEYS TO A SUCCESSFUL DEPLOYMENT

1. Start with a use case tied to a business priority that can show near-term impact
2. Assemble your team and assign responsibilities, with legal and security engaged from the pilot
3. Prepare your data, integrations, and external systems early in your project
4. Design your agent with decisions about its behavior and safeguards as design inputs
5. Roll out in phases with a standing rhythm of measurement and improvement after launch
6. Scale across channels, use cases, and regions using the same foundation as your first agent

ElevenAgents is designed to scale with your agent strategy

Voice and chat agents in 70+ languages, with built-in testing, guardrails, and experimentation. Forward deployed engineers help you design and implement your long-term agent strategy.

Talk to an AI Deployment Expert

Choose the right first use case

If a first use case does not connect to an existing business priority, the program will struggle for attention, resourcing, and expansion no matter how well the agent performs.

Good first candidates tend to share the following properties:

It is a genuine business priority, tied to a cost line leadership tracks, a revenue motion, or a customer experience metric with an executive owner.

The outcomes are measurable. Metrics like resolution rates, containment, satisfaction, and handle time can be compared against the current operation.

Correct behavior is easy to define. The team can say precisely what a good outcome looks like, which makes quality observable and disagreements rare.

The integration surface is bounded. It touches one or two systems of record rather than seven.

There is a clear escalation path. When the agent cannot resolve something, there is an obvious human destination, and the handoff carries full context.

Start with an intent capture analysis to map out the full set of reasons customers contact you and decide which the agent should handle. Your first agent use cases should address common cases which are not too sensitive, where the agent could feasibly outperform the current path. For example, common first use cases in a contact center include intelligent call routing, first-line issue resolution for common queries, and appointment scheduling.

Define your success metrics early, because the definition impacts agent design. For example, an agent built to maximize containment may behave differently from one built to maximize resolution. In general, you should also choose metrics that can be benchmarked directly to current operation, an accurate basis for comparison.

This first deployment not only helps demonstrate the value of a given use case, it also helps build a foundation that every later use case reuses around data readiness, agent configuration, security and compliance, and more.

Assembling your team

INTERNAL RESPONSIBILITIES

One of the strongest predictors of deployment speed is whether the right responsibilities were assigned before the build. The roles below recur across successful programs - one person may hold several, and the right split ultimately depends on your own organization and requirements.

ROLE	RESPONSIBILITY
Executive sponsor	Sets direction, secures resourcing, and unblocks organizational obstacles.
Project lead	Single point of contact who owns the timeline and day-to-day coordination, including user acceptance testing.
Technical agent owner	Defines what the agent should do, what success means, and what to take on next. Manages agent configuration and user access, and owns access to APIs and infrastructure.
Security and compliance lead	Owns the security review and the legal and compliance sign-off, engaged from the pilot.
Knowledge manager	Owns the policy documents and reference content the agent draws on, and the governance of how that content changes.
QA and operations analyst	Tracks containment, accuracy, latency, and escalation behavior, and runs the standing review rhythm.

BUILD, BUY, OR PARTNER

Every team needs to decide which parts of their agent strategy they should build in-house, buy, or partner on.

Some paths we see customers take include:

Hands-on vendor partnership: Deployment engineers embed with the team to design their agent strategy, develop and deploy their agents, and optimize them after launch. This can be a fast route to get a robust agent into production for teams that want expert support throughout.

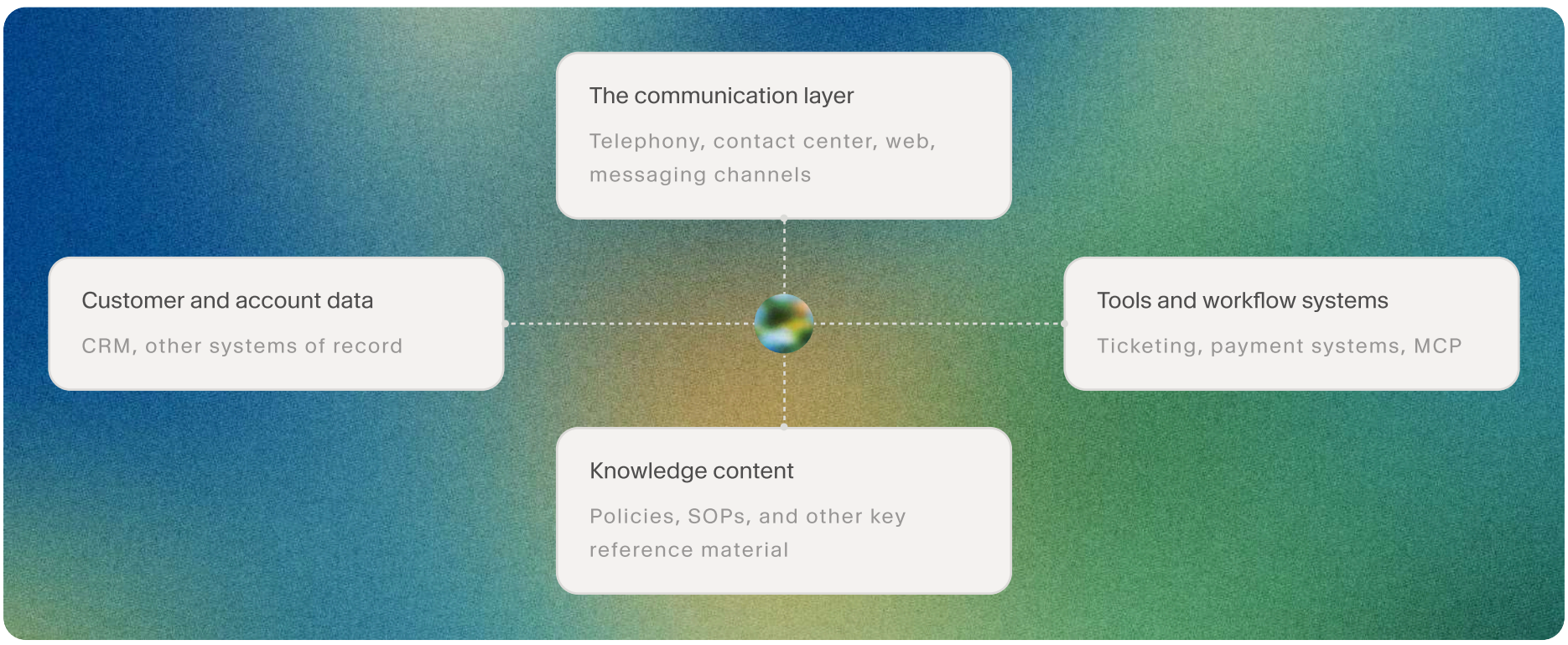
Self-serve build: The team designs, orchestrates, and iterates entirely within an agent platform like ElevenAgents that provides the tooling to configure agents. This suits organizations that would rather build on their own and have some technical resourcing available, but still want to focus on agent design rather than core infrastructure.

In-house orchestration: An internal engineering team owns end-to-end orchestration including RAG and integrations. Platforms like ElevenLabs are brought in only for the voice layer - either Speech Engine which adds voice capabilities to any chat agent, or the underlying speech models via API.

Whichever path you choose, you should make sure your own team has control over the configuration, integrations, and evaluation framework, and can make changes on your own. ElevenLabs offers support for organizations that take any of these paths - from forward deployed engineers who build alongside your team, to a self-serve agents platform, to direct API access.

Prepare your systems and data

An agent's usefulness is bounded by what it can see and do. Deployments often require four types of integrations:



You should kick these off early in your agent build, because they can take longer than expected. For example, connecting to a CRM or contact center platform may require multiple levels of approval. And an early audit of your agent’s knowledge base can prevent issues with contradictory, incomplete or outdated information. At this early stage, you should also decide where data will live - keep it in the cloud, on prem, or limit retention entirely.

Integrations also require escalation paths. An agent that depends on a CRM or order system also needs defined fallback behavior if that system is slow or down like a polite acknowledgment and escalation to a human.

As you’re building out these integrations, you should coordinate closely with relevant teams. For example a support team handling tickets created by the agent or receiving realtime escalations needs to know what the agent does, how their workload will change, and what context arrives with a transferred conversation.

REAL-WORLD EXAMPLE

30+ backend actions for agentic shopping and support

Rohlik, one of Europe's leading e-grocers, built an agent on ElevenAgents for agentic shopping and support for 3 million customers. It connects to Rohlik's backend systems via Model Context Protocol (MCP), enabling it to take more than 30 actions - checking real-time order status, making live modifications to existing orders, and resolving delivery issues autonomously.

30+ actions available in backend systems	4 different channels deployed
2x faster resolution rates	50% reduction in operational costs

Building your agent

The technical build is covered in the Playbook series - prompt and tool design (Volume 1), multi-agent orchestration (Volume 2), and safety guardrails (Volume 3). This section is about the layer around it: the governance decisions that shape what gets built, and the testing that proves it works.

DECISIONS THAT SHAPE THE BUILD

Beyond the formal engineering work, your team needs to make a series of decisions about your agent’s behavior.

For example:

- Should it sound warm and conversational, or crisp and professional?
- Which actions require verification first, like confirming identity?
- What topics should it refuse or route to a human?

These are your team's calls to make - and to revisit, since policies and products change and the agent has to change with them. Look for a platform that lets you configure each one and adjust it later on your own; ElevenAgents gives you that control, from voice and personality to guardrails and escalation paths.

TESTING AND PRE-LAUNCH VALIDATION

Run your agent through simulated conversations to find issues before they reach a user. The same exercise should check both normal paths and adversarial ones. A comprehensive starting suite draws on your system prompt instructions and guardrails, your knowledge base, and real conversations from the current operation.

Testing should happen before the initial launch and any later change. As your agent encounters new scenarios in production, any failures can be folded back into the test suite so the same problems don’t recur.

Testing also helps earn buy-in from security, legal, compliance, and risk. The fastest programs engage them during the pilot. For regulated enterprises, independent standards raise the bar further: ISO 42001 governs the AI management system behind the agent, while AIUC-1 puts the agent itself under adversarial pressure across thousands of scenarios, with periodic re-testing by outside evaluators. ElevenLabs was the first company to earn AIUC-1 certification for voice agents and helped shape its voice-specific requirements. Because of the safeguards are built into ElevenAgents, agents built on the platform can achieve certification in weeks.

REAL-WORLD EXAMPLE

AIUC-1 certification for a customer-facing property agent

Immobiliare.it, Italy's largest property platform, built a conversational voice agent on ElevenAgents to handle property inquiries around the clock. With customer trust on the line, security and safety were non-negotiable, and the team wanted independent validation beyond their own testing.

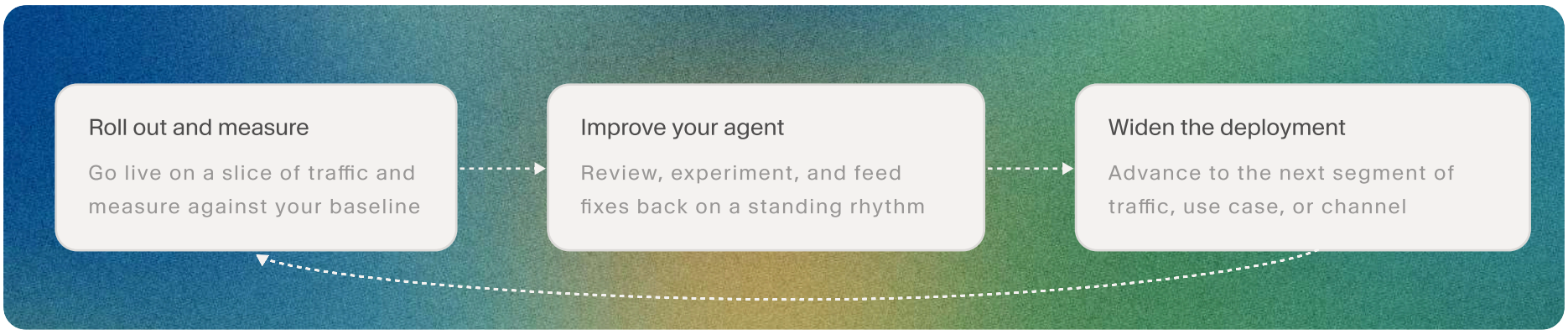
They took the agent through full AIUC-1 certification, running it against thousands of adversarial scenarios. With this testing, Immobiliare.it got third-party validation of their agent's security controls, found and fixed vulnerabilities that internal testing had missed, and built the confidence needed to deploy their agent.

4 weeks for full testing and certification

Thousands of scenarios tested

Roll out in phases

No amount of pre-launch testing fully predicts production, so the pilot should limit exposure while the system proves itself. Then a cycle repeats: roll out to a small cohort, measure, improve your agent, and widen the deployment - with each new intent or segment re-entering the cycle from the start.



Roll out and measure

Start off by piloting the agent with a small slice of real traffic - a set time of day, a single region, or one segment. This way, you learn from live conversations while mitigating risk. Measure the metrics you defined earlier, like resolution, escalation accuracy, and customer sentiment, against the baseline from the current operation.

Each increase in traffic should clear a gate set beforehand: written pass criteria, a named owner who makes the call, and a rollback path that can revert if the numbers turn. Only roll out to the next cohort of traffic when the given criteria is met.

REAL-WORLD EXAMPLE: **FOUR-PHASE ROLLOUT**

A European postal service began with a contained pilot: voice agents handling parcel-delivery calls only outside business hours. This was relatively low risk traffic but it still had value as there previously was no after hours coverage.

Once it met the pre-established success criteria, they scaled through 3 more phases: More complex workflows, a partial daytime rollout with human handoffs, and finally full production across several countries and languages.

Improve your agent

The deployments that pull ahead are the ones that keep improving after launch.

That improvement runs on a standing rhythm:

- **Review conversation outcomes** to create a prioritized list of experiments
- **Ship changes as A/B tests** against live traffic, so each one is measured against the status quo
- **Expand your test suite** with issues found in production, so the same problem doesn't recur

REAL-WORLD EXAMPLE: **A/B TESTING**

3Shape, a global dental-imaging company, ran A/B tests across its regional deployments, varying voices, first messages, and escalation triggers by market. The result was reduced call abandonment by over 50%, achieved through a steady cadence of measured changes.

50% reduction in call abandonment rate

Scale across the business

Once the first deployment is stable, growth comes from scaling to more traffic, segments, capabilities, channels, and markets. Many of the foundations from your original deployment carry over, so the economics tend to improve. Each vector carries its own considerations - some add little new risk while others warrant their own gated launch.

SCALING VECTOR	CONSIDERATIONS
Traffic volume	Confirm capacity, latency, and monitoring hold as load rises; overall agent configuration remains unchanged
Customer segment	Add eligibility rules and potentially new logic for higher-stakes cohorts; most of the agent configuration is similar
Capability	New action permissions, guardrails, and fresh sign-off to move to handle more complex workflows, building more functionality into the same agent
Use case	Add new intents, knowledge, and success metrics - handled by the existing agent or a new one; integrations, evaluation, and governance generally carry over
Channel	Add channel integration, surface-specific delivery instructions and testing; agent logic and knowledge generally stay the same
Region and language	Consider new voices with local accents, QA with native speakers, and check compliance with local regulations; overall agent design holds, and voices can be easy to swap

REAL-WORLD EXAMPLE

From one use case to the core of sales at a used-car marketplace

CARS24, one of India's largest used-car marketplaces, runs more than 300 retail hubs, with 1,300+ contact center agents handling nearly half of all sales by phone. They started with one small use case: missed-appointment reminders - short calls where a mistake cost little - and proved the model on a small slice of live traffic before widening.

Each new deployment then followed the same ramp: 5% of traffic, then 10%, 20%, 50%, and 100%, with a two-day hold at each step to confirm the metrics held before expanding. That approach let them scale along every axis, with the foundation built for those first simple calls carrying through.

Today their agents run live negotiation, discovery, and financing across 13 languages - automating a quarter of all call volume while assisting nearly half of sales.

45% of sales assisted by AI voice agents

25% of call volume automated

50% lower calling costs

3M+ minutes of AI-supported calls

A deployment readiness checklist

Below is a checklist for a first deployment - setting a foundation for success and scale.

1. Scope

- One use case tied to a real business priority, with a clear escalation path
- Correct behavior easy to define and observe
- Integration surface limited to one or two systems of record
- Deferred use cases added to a backlog

2. Team

- Executive sponsor, project lead, and agent owner named, with time allocated
- Technical lead with authority to grant system access; security and compliance owner named
- Knowledge content owner and QA analyst identified for post-launch operation

3. Systems

- API access requests submitted early on
- Relevant knowledge base content audited
- Fallback behavior defined for when an integrated system is slow or down
- Receiving teams briefed on escalation design and context handoff

4. Agent design

- Guardrails, escalation paths, and primary metrics specified as design inputs
- Evaluation criteria running on every conversation from day one
- Legal and compliance engaged during the build

5. Rollout and operation

- Establish go or no-go gates with written criteria and rollback paths and with named owners
- Start on a small slice of traffic, expanding as each gate passes
- Consistent monitoring with a standing review rhythm and experiment pipeline

6. Scale

- Roll out new use cases, building on a shared foundation - integration, evaluation, governance
- Each new market or channel run can have its own gated launch
- Native-speaker QA for each new language, calibrated against automated evaluation

ElevenAgents is designed to scale with your agent strategy

Voice and chat agents in 70+ languages, with built-in testing, guardrails, and experimentation. Forward deployed engineers help you design and implement your long-term agent strategy.

Talk to an AI Deployment Expert

1. Deloitte, "State of AI in the Enterprise: The untapped edge" (2026)