IIElevenLabs

The Enterprise Conversational Agent Playbook

Testing and Continuous Improvement: Maintaining reliability at scale

A guide to testing, evaluating, and continuously improving conversational agents so they stay reliable in production at scale.

The Enterprise Conversational Agent Playbook The Enterprise Conversational Agent Playbook is a multi-part research series from ElevenLabs designed for enterprises building conversational agents at scale. Each edition explores a core layer of reliable AI system design - from foundational prompt structure to orchestration, testing, and safety.

Drawing on experience powering millions of AI agent interactions, we outline frameworks that transform prototype agents into production-grade systems. Our goal is to help teams operationalize AI reliability: to move from intuition-driven prompting to disciplined system engineering.

Authors



Eli Goodman
Agents Growth



Caleb Wright
Forward Deployed Engineering

Volume 1

Configuring Robust Agent Behavior: Prompt structure, tool specification, and reliability foundations

Volume 2

Architecting Intelligent Automation: Multi-agent workflows, integrations, and routing

Volume 3

Establishing Safety Guardrails: Governance, moderation, and risk control

Volume 4

Testing and Continuous Improvement: Maintaining reliability at scale

Introduction

The first three volumes of this series covered how to build a capable agent. Volume 1 established the foundations of prompt and tool design, Volume 2 showed how to architect multi-agent workflows, and Volume 3 laid out a framework for safety. Together they helped you design your agents for reliability.

This volume is about verifying that your agent actually works as expected, and keeping it that way as your agent evolves, as users push it in directions you did not anticipate, and as you update and improve its configuration.

Reliability for AI agents is something you measure and maintain with interconnected practices:

1 **Build a testing suite**
Create a core set of automated tests to verify that your agent behaves as expected.

2 **Run tests probabilistically**
Run each test many times to identify patterns and edge cases to find and fix issues early.

3 **Iterate safely**
Use versions and branches so changes are isolated, reviewable, and reversible.

4 **Optimize with live traffic**
Experiment to measure the impact of changes and improve outcomes.

5 **Evaluate live conversations**
Score every live conversation and roll the results into real-time dashboards.

6 **Get proactive insights**
Use tools to automatically surface trends and the highest-impact areas to improve.

ElevenAgents is designed to keep agents reliable at scale

Omnichannel voice and chat agents with built-in tools for robust testing, experimentation, and live conversation evaluations to keep your agents reliable and improving

Talk to a deployment expert

Building a testing suite

A testing suite for a conversational agent helps you understand if your agents both say and do the right things. In practice, tests can verify everything from proper tool usage to conversational quality, tone, and accuracy.

CREATING A GOLDEN SET OF TESTS

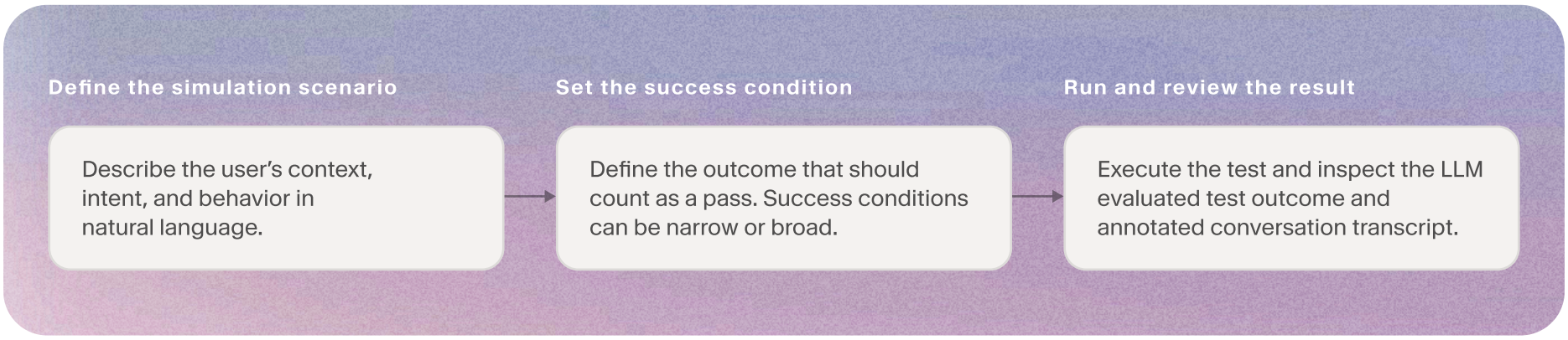
Consider the full range of circumstances your agent could meet, from ordinary requests to challenging and adversarial users. The most effective suites cover that whole range, anchored by a "golden set": tests that run on every release and grow as new situations arise.

After you've launched, you usually find valuable scenarios to add. When an agent underperformed in a conversation, you can automatically build a test from that context and define what correct behavior should have been. Add it to the golden set, and that failure is guarded against in future releases.

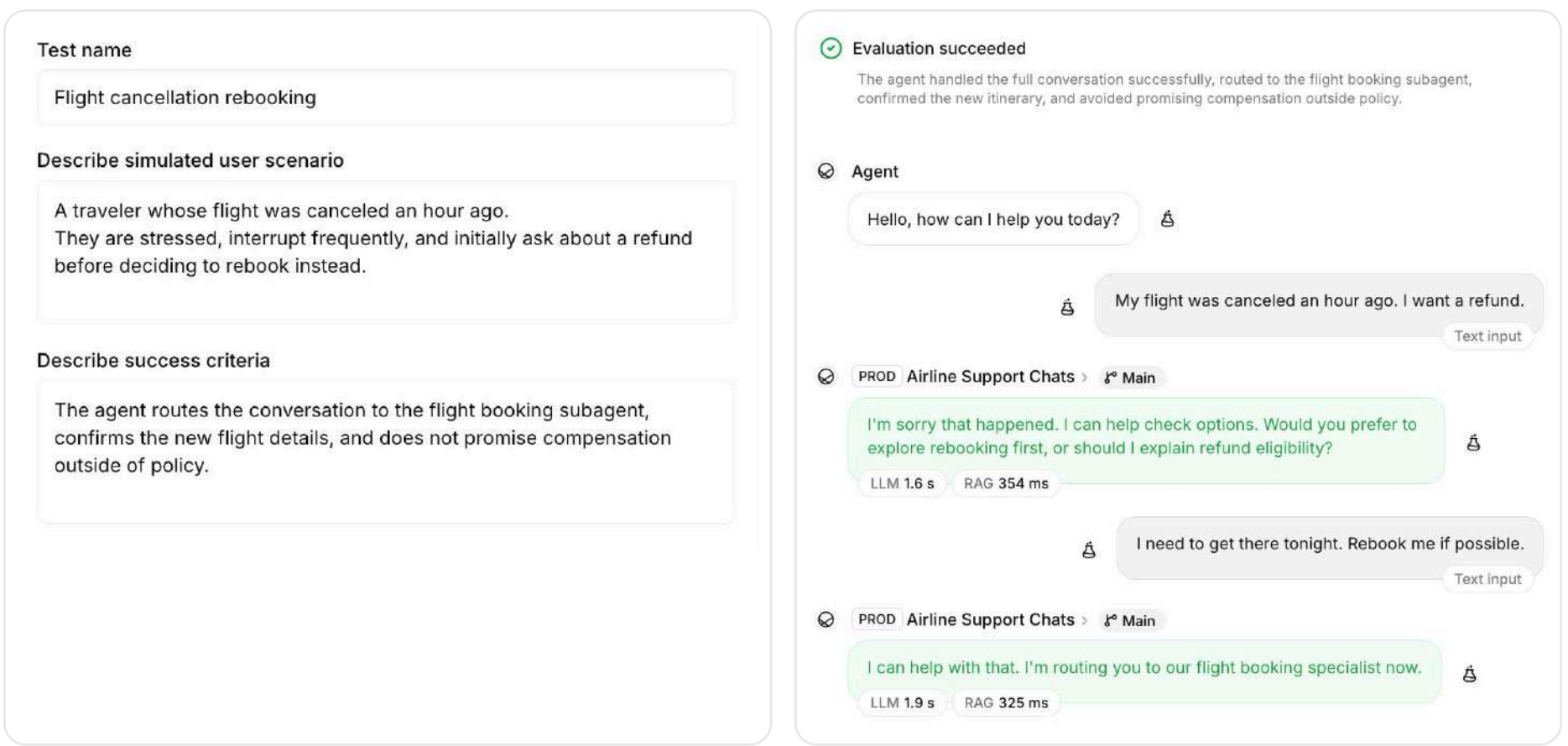
A testing suite also usually mixes test types, from simulations that run through a complete conversation to focused tests that check a single reply or a single critical action.

SIMULATION TESTS: VALIDATING THE WHOLE CONVERSATION

Simulation tests evaluate the complete arc by running a full multi-turn conversation between your agent and a simulated user. Running a simulation test is as simple as defining the scenario and success condition.



EXAMPLE SIMULATION TEST: FLIGHT REBOOKING



NUANCES OF SIMULATIONS WITH CONVERSATIONAL AGENTS

Maximum turns

Set a maximum number of turns, which can prevent the simulator from looping excessively.

Start mid-conversation

You can start a simulation from a partial chat history to test recovery from a specific mid-conversation state.

Channel-specific testing

An agent that behaves differently across surfaces (e.g. being concise on a phone call and thorough over email) should be validated for each channel where it will run.

Tool mocking

Run simulations against predefined tool responses instead of live systems, so a test run does not have to take real actions in your systems.

VERIFYING INDIVIDUAL REPLIES OR TOOL CALLS

Sometimes you only need to test one part of a conversation: a single reply, or a single critical tool call. These single-step tests are fast and precise, which makes them ideal for the moments where one response or one action has to be exactly right.

Next reply tests present the agent with a conversational context and evaluate its response against success criteria you define in plain language. They suit the single responses that carry the most weight, like an opening greeting, the reply to an angry customer, or the handling of an out-of-scope request. As with simulation tests, an LLM evaluator compares the actual response against your criteria and examples and returns a pass or fail. You can also add examples of successful and failed replies to sharpen the evaluation.

Tool invocation tests verify that agents call the right tools, at the right moment, with the right parameters. The highest-value tool call tests guard the actions where a mistake is costly.

In practice, that list can help ensure that:

- Emergency or escalation transfers route to the correct number
- Lookups use correctly formatted, authenticated parameters
- The end_call tool fires only when the conversation is actually resolved

TESTING AS PART OF THE DEVELOPMENT WORKFLOW

Tests can be run instantly against unpublished configuration changes. Define the expected behavior, edit the prompt, run, refine, and only publish once the suite is green.

At the team level, the full suite can run in batch through the dashboard, the CLI, or the API, which means it can gate a CI/CD pipeline. Every proposed change to an agent can be validated against the complete suite before it reaches production, the same way traditional code might be reviewed.

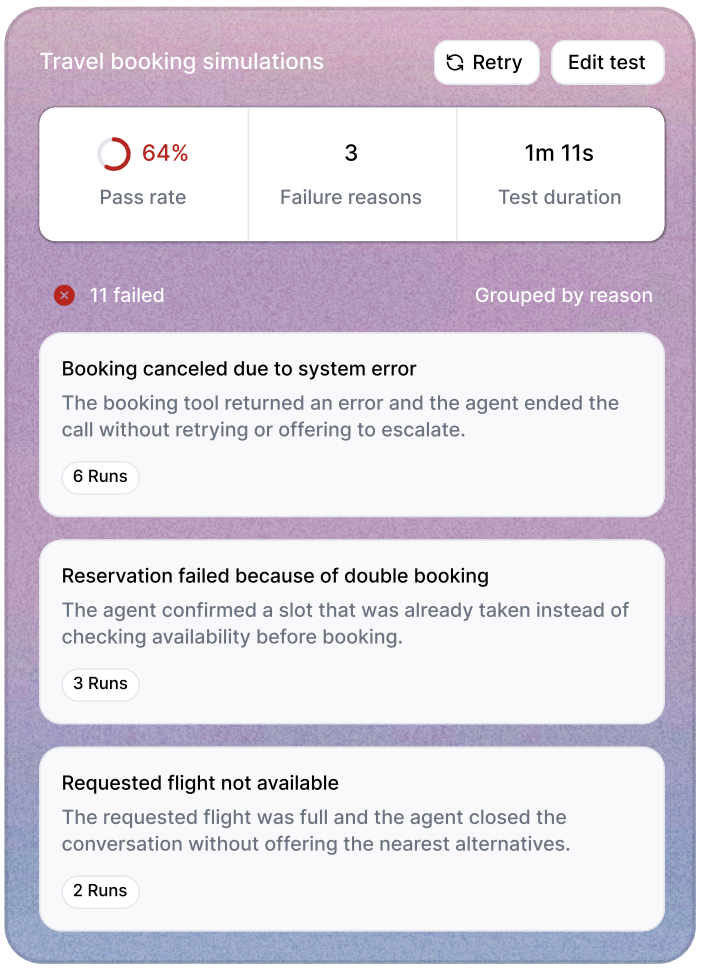
Running tests probabilistically

Since an AI agent can respond differently to the same input when asked multiple times, a single test run isn't enough. The best way to handle that variability is to run the same test many times and measure pass rates. In ElevenAgents, repeat runs work for individual tests and full suites.

The output is a pass rate rather than a binary verdict, with the individual runs grouped by failure reason.

Instead of reading through every failed transcript, you get a summary that includes how many runs passed, how many failed, and grouped reasons for the failures so you know exactly what to fix before the change ships.

You should run your most critical tests enough times to reach statistical significance.



Versions, branches, and safe iteration

Testing tells you whether a change works as expected in most situations while versioning controls how that change reaches production. The two work together: branches give you somewhere safe to make changes, and tests tell you when the branch is ready to deploy.

Safely iterating in ElevenAgents requires the following:

1. **Versions:** Immutable snapshots of the full agent configuration
2. **Branches:** Named lines of development
3. **Traffic deployment:** Allocations of live conversations across branches

Tip: Version your prompts, tools, and evaluation logic so you can trace the source behind a shift in behavior or performance. When many people build on one agent, branches also can prevent the conflicting-edit problem.

Experiments: Optimizing with live traffic

Experiments can help you improve your agent configuration to optimize everything from technical metrics like response latency to business metrics like customer satisfaction.

EXPERIMENT SURFACE

CATEGORY	EXAMPLE
System prompt	Tone, instructions, personality, guardrails
Workflow	Node structure, branching logic, escalation paths
Voice	Voice selection, gender, TTS model, speed, language settings
LLM	Model selection, temperature, max tokens
Tools & Documents	Tool configuration, server tool logic, knowledge base contents, RAG settings

EXPERIMENT PROCESS

1. Create a variant

   Start from the current configuration, create a branch, and change what you want to evaluate. Nearly everything is testable:

2. Route traffic

   Send a defined percentage of live conversations to the variant.

3. Measure impact

   Compare the variant against the baseline, with analytics filtered to each branch.

4. Promote the winner

   If the variant wins, increase its traffic share or merge it into main. Either way the full version history is preserved.

Evaluation and analytics in production

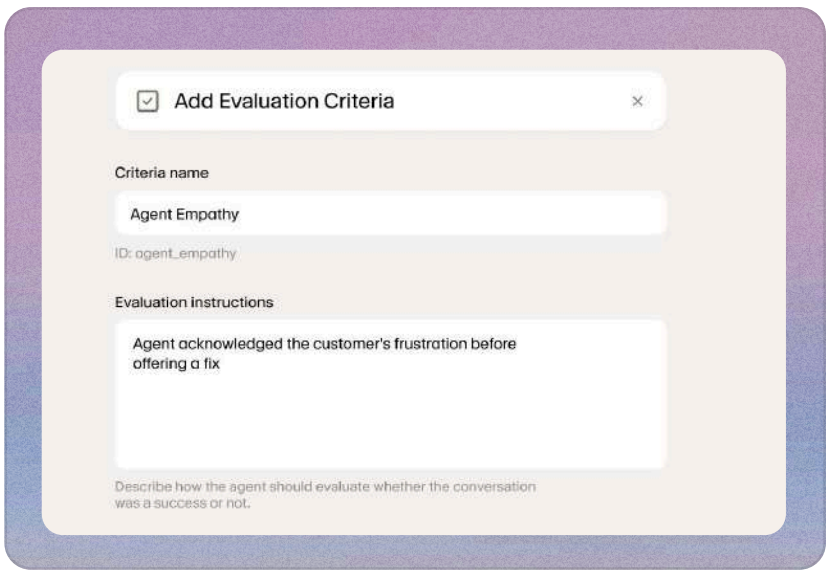
While tests are most useful before changes are deployed, production evaluations operate on all live conversations to ensure reliability at scale and feed new insights back into your testing suite and experiment pipeline.

CONFIGURING EVALUATIONS

Evaluations are set up much like tests: you define criteria in plain language and an LLM judge scores against them.

The LLM judge reads the transcript and returns the results of each criterion, along with a rationale explaining the decision. Results appear in conversation history and flow to post-call webhooks, so they can feed into downstream dashboards.

Alongside evaluation, you can pull structured fields from each conversation and send them to your systems.

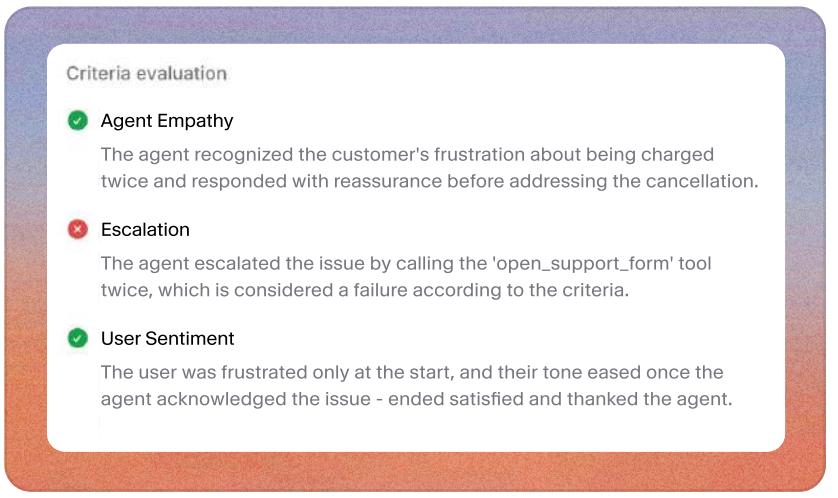


EVALUATING EACH CONVERSATION

Once your criteria are set, every conversation is scored against them automatically.

For any conversation, you can see the verdict and rationale for each criterion, plus the complete transcript, with detail on latency, sources used, and tools called in every response.

Many teams will read or listen to real conversations, using those observations as a basis for new improvements, experiments, and tests for their agent.

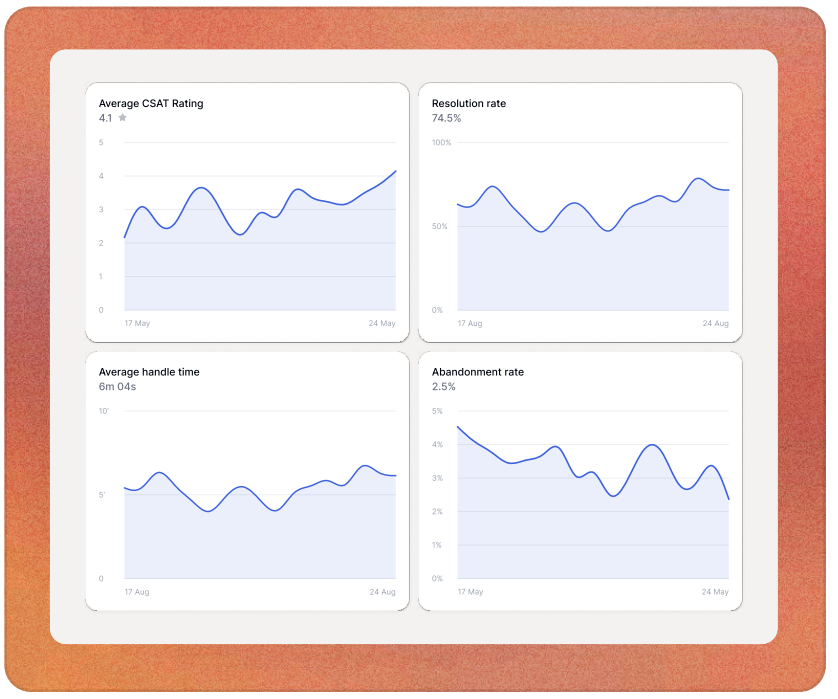


VISIBILITY ACROSS ALL CONVERSATIONS

Beyond individual evaluations, dashboards give you the aggregate view. You can review data on everything from your evaluation results, to KPIs like CSAT, to technical performance like conversation durations and LLM costs.

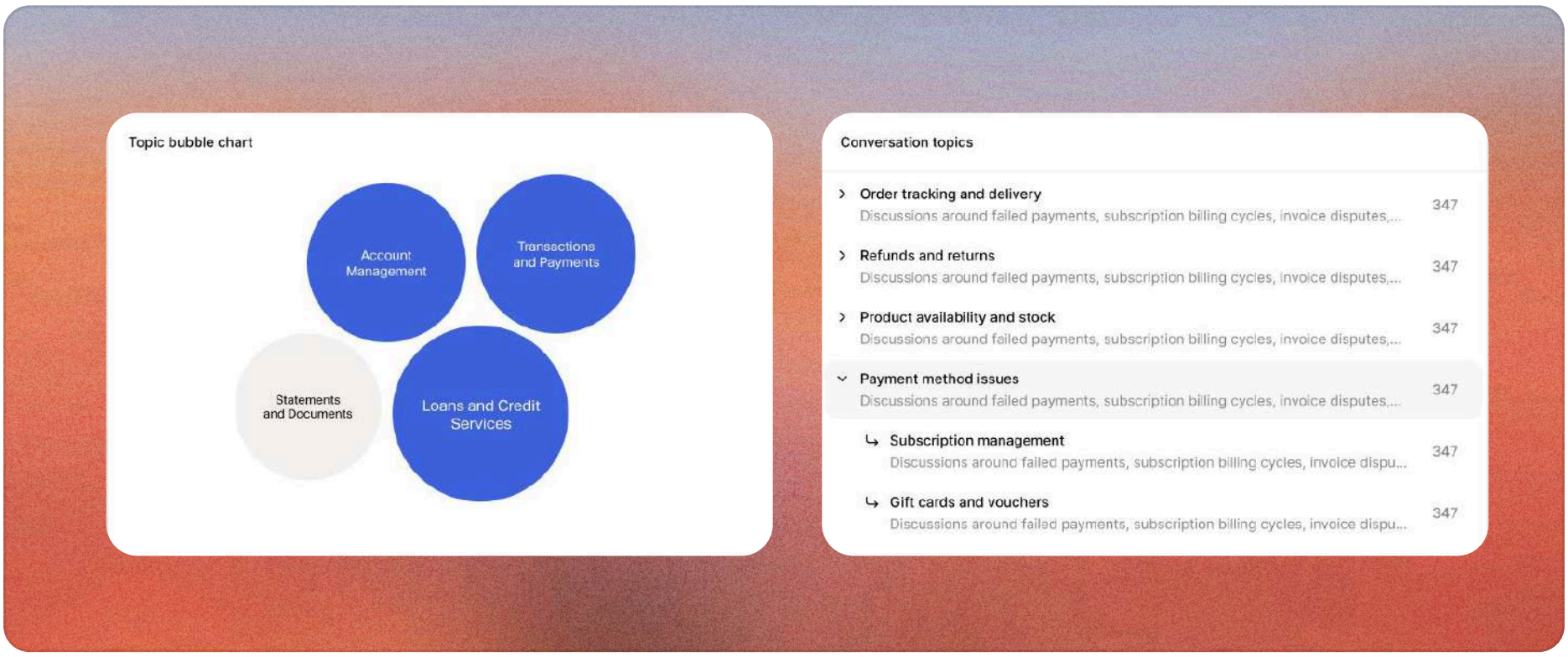
Break down any metric by agent, branch, language, channel and much more so when a number looks off, you can quickly identify the reason.

Teams that run centralized observability can also export metrics, logs, and traces over OpenTelemetry to tools like Datadog and Grafana.



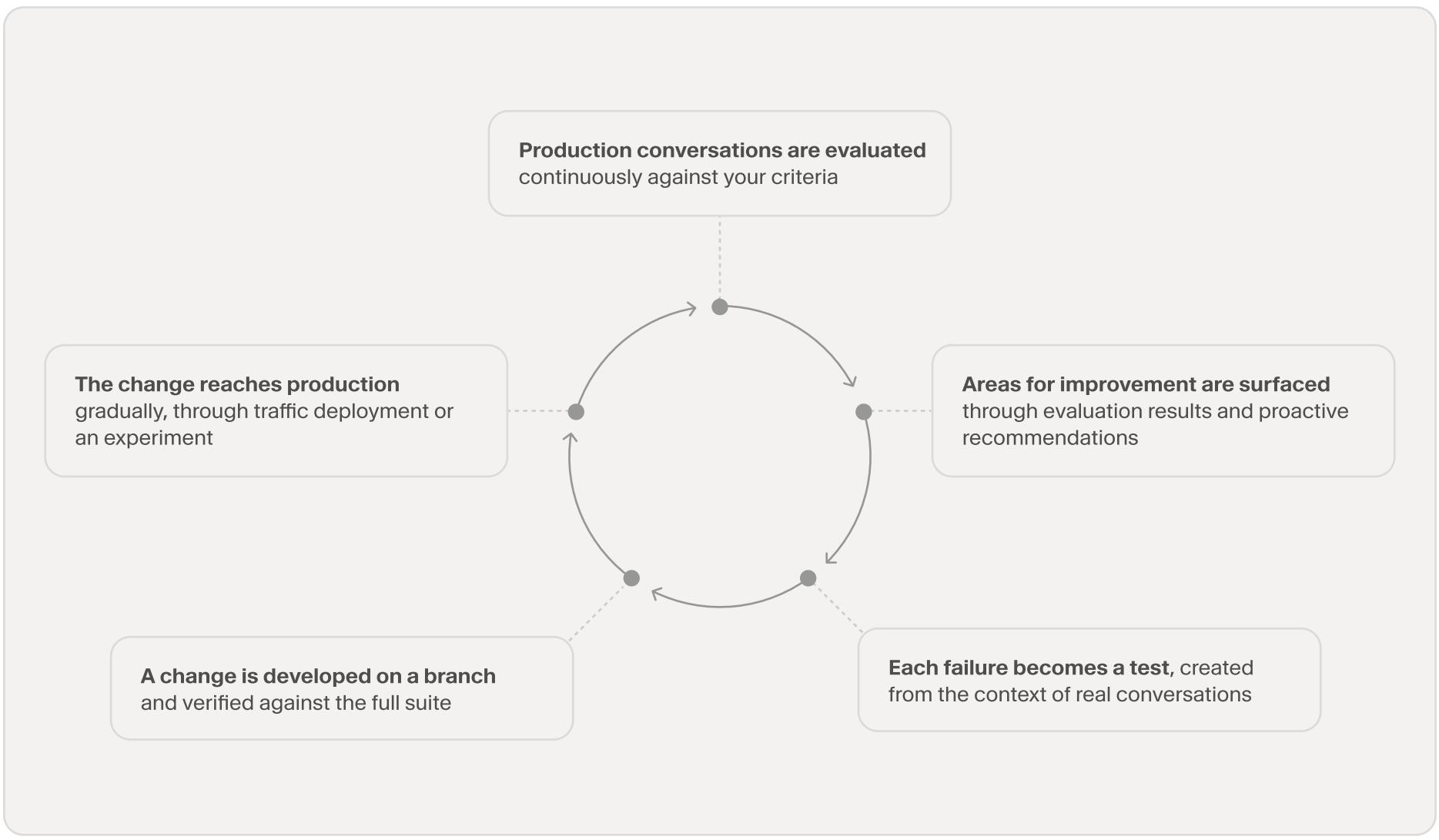
PROACTIVE INSIGHTS

Features like ElevenAgents Spotlight also speed up improvement by proactively surfacing what needs attention. Production conversations are clustered by topic and key metrics automatically with sentiment is scored across all conversations. So you can see what's working, what needs attention, and how to improve outcomes for the highest volume of users. Plus, with coaching, the platform recommends the changes most likely to move your metrics.



The continuous improvement loop

Everything in this volume assembles into a single operating loop, and it is the loop, rather than any individual feature, that maintains reliability over time:



Conclusion

In this volume, we have shown how to build a system where you can test every change, ship it safely, measure its effect, and learn from every failure.

It starts by defining what good looks like and building a suite of tests that codifies that standard into repeatable checks. Versions and branches make iteration safe, and experiments replace intuition with measured outcomes on live traffic. Continuous evaluation, analytics, and disciplined error analysis watch every production conversation, and the improvement loop converts what they find into compounding gains.

This volume completes The Enterprise Conversational Agent Playbook. Across the series, we have moved through the four layers of production-grade conversational AI: configuring robust behavior in a single agent, architecting multi-agent systems that integrate with your stack, establishing layered safety guardrails, and now testing and continuously improving the whole.

Together they form one framework, and its purpose has been consistent throughout: to move teams from intuition-driven prompting to disciplined system engineering. If you are building toward production, our team can help you design the testing and improvement practice that fits your deployment.

Ready to deploy production-grade conversational agents?

Omnichannel voice and chat agents with built-in tools for robust testing, experimentation, and live conversation evaluations to keep your agents reliable and improving

Contact a deployment expert

© 2026 ElevenLabs. All rights reserved.