ancha.

Production, simulated.

Stateful testing playgrounds for AI agents.

Your agent works inside a simulated company. Every tool keeps its state, call after call, and you judge each attempt by what it changed there, not by what it said.

Slack GitHub Datadog PagerDuty Salesforce HubSpot Gong GA4 Google Ads LinkedIn Ads ChatGPT Ads

Where agents get tested today.

Mocks forget.

A stubbed Slack returns the same reply no matter what the agent did one call earlier.

Staging is shared.

One workspace, many runs. Yesterday's messages and tickets leak into today's result.

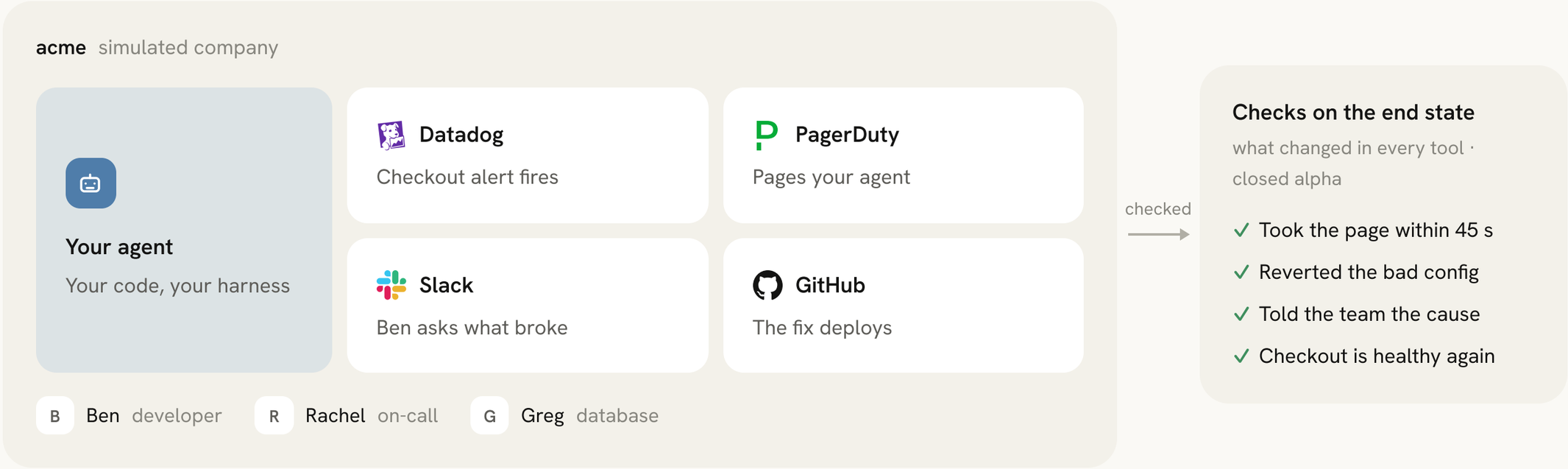
Production is production.

The first real test of a refund or a page is a customer, a teammate, or an on-call engineer.

Ancha gives every attempt its own copy of the company, with each tool's state intact, and shows you what the attempt left behind.

A stateful replica of your agent's entire world.

The tools it calls, the people it works with and the things that happen around it, copied for each attempt and sharing one state.



One clock, deterministic ids, per-world credentials. Nothing an attempt does can reach the real Slack, GitHub or a customer.

Tests that remember what your agent did.

Every call changes what the next one sees. When the attempt ends, you read the world back, tool by tool.

Before the attempt starts

Checkout error rate monitor	alert
Incident: checkout failing	triggered
#ops · Ben: “Which deploy was this?”	unanswered
PR #88 · revert checkout config	open

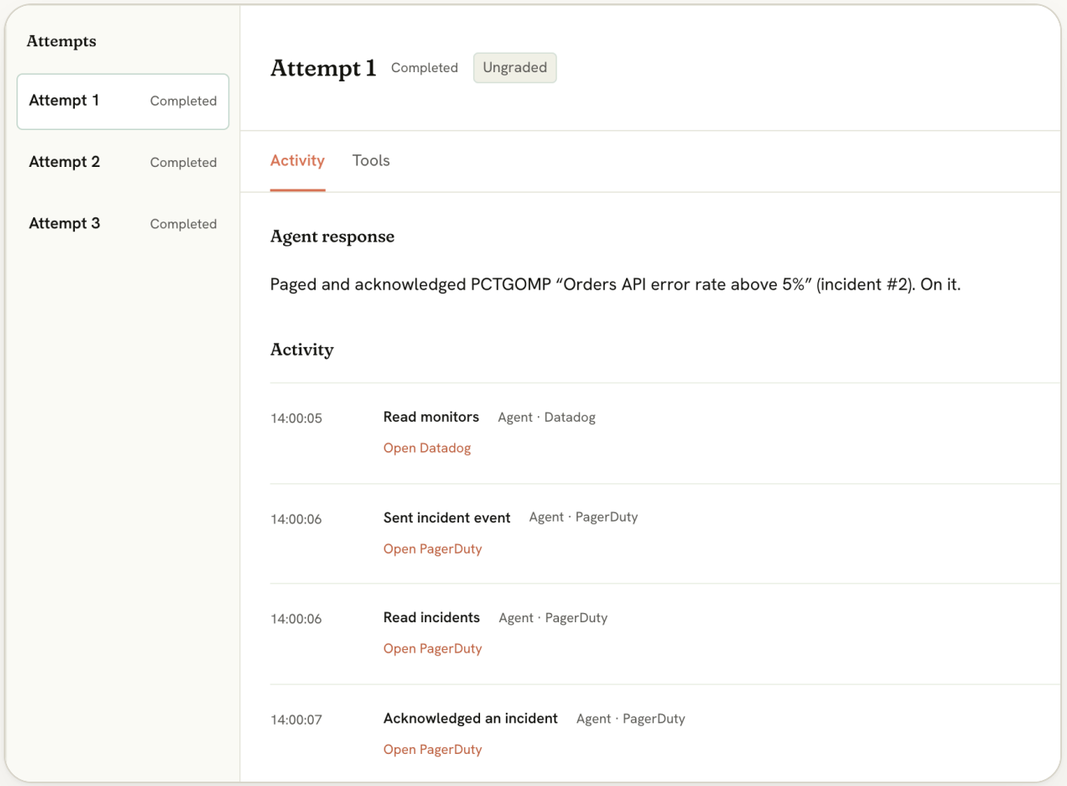
After the attempt ends

Checkout error rate monitor	ok
Incident: checkout failing	acknowledged · 38 s
#ops · agent: “Config push at 14:02. Reverting.”	reply
PR #88 · revert checkout config	merged

An attempt is judged on the right-hand column, read back from every tool, not on its transcript.

Same start. Independent attempts. Every call kept.

A run from the demo workspace: an on-call agent pages and acknowledges an incident, three times from one seed.



Screenshot from app.ancha.io. Checks are in closed alpha, so attempts show Ungraded.

Three attempts, one start

Each attempt gets its own copy of the world. What one attempt changes, no other attempt sees.

Every call, in order

Reads, writes and messages, each with its tool and the simulated time.

Then the end state

The Tools tab reads back what the attempt left in every twin.

What teams put in a playground.

Four agents from the demo workspace, each a saved test with tasks and repeated attempts. Bring your own agent; these show the shape.

Support triage

Slack · GitHub

- File the latest question as a GitHub issue
- Post the workaround
- Reply with an ETA
- Daily summary

4 tasks · 9 attempts

Incident response

Datadog · PagerDuty · Slack

- Page and acknowledge
- Mute during rollback
- Drill: page and resolve
- Status report

4 tasks · 9 attempts

CRM hygiene

HubSpot

- Add a contact
- Advance the Acme deal
- Move to a named stage
- Pipeline report

4 tasks · 9 attempts

Release notes

GitHub · Slack

- Notes to the team
- Notes to #announcements

2 tasks · 5 attempts

Each test runs against its own world: Support workspace, On-call, Sales CRM and Engineering.

Fits the way you already build.

Point your agent at a hosted world from anywhere, or hand Ancha the agent and let it run the attempts.

Sandbox the world is hosted, your agent runs anywhere

```
$ npm install -g @ancha-labs/cli
$ ancha login
$ ancha sandbox create --tools slack,github
$ ancha sandbox connections <id>
$ ancha sandbox fork <id> --count 20
```

```
SLACK_API_URL=https://app.ancha.io/t/…/e/…/slack/api/
SLACK_BOT_TOKEN=xoxb-…
GITHUB_API_URL=https://app.ancha.io/t/…/e/…/github
GITHUB_TOKEN=ghp_…
```

Playground run your agent in its own container, N attempts

Upload the folder or point at a repository at app.ancha.io

Pick tools, write tasks, choose attempts results grouped by task

```
$ ancha secrets set OPENAI_API_KEY
```

read from your shell, stored on your workspace's machine

`ANCHA_API_KEY` one key for scripts and CI, no browser

The tools, as twins.

The endpoints your agent already calls, over one shared state, with the credentials it expects. A call a twin does not support fails in the service's own error format, never as a fake success.

Slack channels, threads, events	**GitHub** repos, PRs, checks, webhooks	**Datadog** monitors, logs, metrics	**PagerDuty** incidents, on-call, routing keys
Salesforce closed alpha	**HubSpot** closed alpha	**Gong** closed alpha	**GA4** closed alpha
Google Ads closed alpha	**LinkedIn Ads** closed alpha	**ChatGPT Ads** closed alpha	**Another tool?** tell us which one

Next Stripe Linear Notion Gmail Okta Shopify Attio Better Stack

Your agent reaches the world and your model. Nothing else.

Two containers per attempt on a private network. Nothing an attempt does can touch the real Slack, GitHub or a customer.

Agent container

your project, your start command

- Read-only root, tmpfs workspace · fresh per attempt
- Task and twin credentials as environment · never in the prompt
- Model traffic only to the exact domain you allow · openrouter.ai, api.openai.com

private →

World container

the twins, one shared state

- Per-world fake credentials · useless anywhere else
- Deterministic clock and ids · same seed, same start
- State captured at the end · the evidence for the verdict

Local: Docker on your machine, nothing leaves it	ancha ui · ancha run
Hosted: one private machine per account, worlds reachable from your CI	app.ancha.io

Bring your agent.

A project folder and a start command are enough. The first world is ready when you sign in.

1

Sign in

Email link or password at app.ancha.io. Your workspace starts with a sample agent, a world and a test you can rerun.

2

Connect your agent

Upload the folder or point at a repository. Ancha prepares it in a container and checks it can reach the twins.

```
$ npm install -g @ancha-labs/cli
$ ancha login
```

3

Run, read, repeat

Pick tools, write tasks in plain English, choose how many attempts. Open any attempt and read the world it left behind.

Sign in app.ancha.io Design partners ancha.io