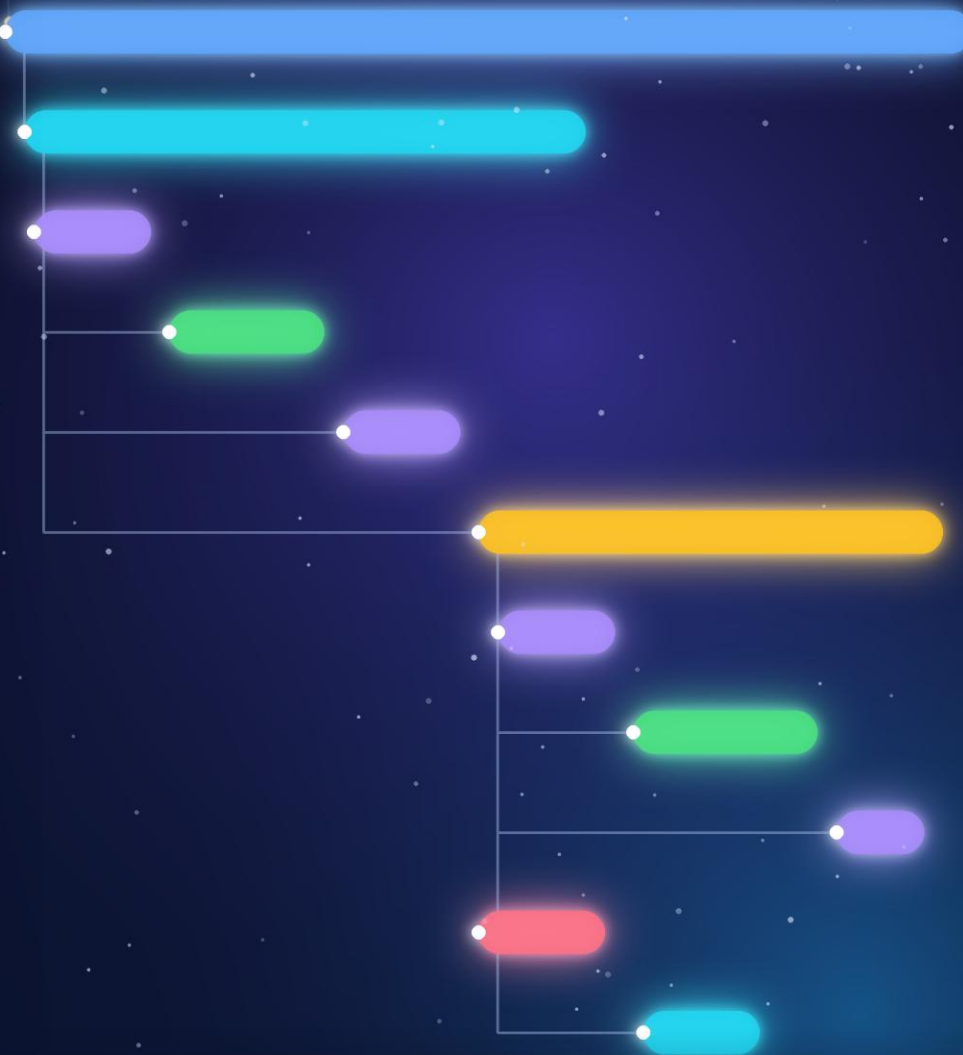


ESTUDO · AGENTES DE IA

Rastreamento de Execução de Agentes de IA

Fundamentos, estado da arte e comparação de 13
plataformas

Aridio Silva @aridiosilva
Outubro 2026



Resumo: o que o estudo conclui

- **Rastreável, mas sem padrão estável.** Todas as grandes plataformas já permitem rastrear cada passo de um agente; nenhuma oferece um padrão estável.
- **Trace não prova acerto.** Mostra o que o agente fez, não se a tarefa deu certo: é preciso avaliação separada.
- **Maturidade em andamento.** Vários recursos estão em beta ou preview; as convenções OTel GenAI seguem em Development.
- **Privacidade varia.** Prompts e ferramentas são capturados por padrão em algumas plataformas e não em outras.

13

plataformas comparadas: 5 fabricantes e 7 independentes

47

referências com validação [A], [B] ou [C]

10/10/2026

data do levantamento; versões lidas no PyPI e no npm

1

Por que rastrear agentes de IA

- **O que o operador precisa saber:** se concluiu a tarefa, onde errou ou se desviou, quanto tempo e custo consumiu e quais passos executou.
- **Comportamento não determinístico:** agentes decidem em tempo de execução quais ferramentas chamar e a quem delegar.
- **Logs lineares não bastam:** a resposta é adaptar o tracing distribuído ao ciclo do agente.
- **Quem fornece o registro:** a camada que executa o agente (SDK, framework ou plataforma), não o modelo sozinho.

Por que importa: Sem rastreamento, não há como depurar onde o agente falhou, medir latência e custo por etapa nem auditar o que foi feito.

QP1

Como trace e span se aplicam à execução de um ou mais agentes?

QP2

Como responder a conclusão, erro, desvio, duração e passos?

QP3

Quais plataformas oferecem isso, com que maturidade e lacunas?

Trace, span e sessão

- **Span:** trecho rastreado da execução; registra uma etapa específica do trabalho do agente.
- **Trace:** a jornada completa de uma operação, composta por spans.
- **O que um span carrega:** início e fim (duração), span pai (hierarquia) e, onde há suporte, status e eventos.
- **Sessão:** agrupa vários traces de uma mesma conversa ou interação.

Por que importa: spans individuais permitem medir tempo, tokens e custo de cada etapa e entender o fluxo de decisão do agente.

Tipos mais comuns de span

Chamada ao modelo

tempo, tokens e, conforme a plataforma, o prompt enviado

Uso de ferramenta

qual ferramenta, os parâmetros e o resultado

Handoff

transferência de controle para outro agente

Três nuances ao ler um trace

1

Trace nem sempre é a tarefa inteira

- OpenAI Agents SDK: o trace é uma operação ponta a ponta; group_id agrupa execuções
- AgentCore: o trace é um único ciclo de requisição e resposta; a sessão agrupa vários traces

2

Handoff nem sempre é um tipo de span

- OpenAI tem span dedicado de handoff
- No OTel, o subagente aparece como outro invoke_agent aninhado
- A Microsoft acrescentou um span para a comunicação entre agentes

3

Há mais spans que modelo, ferramenta e handoff

- Existem spans de guardrail e de turno
- O Claude Agent SDK tem um span filho para o tempo em que a ferramenta aguardou aprovação do usuário

Por que importa: cada plataforma modela trace, sessão e handoff de um jeito; confira o modelo de dados antes de comparar números.

Árvore de spans: reservar passagem



- **Árvore, não lista:** cada passo é um span aninhado dentro do trace.
- **No OTel,** o handoff aparece como subagente aninhado (invoke_agent).
- **No OpenAI Agents SDK,** o handoff teria span próprio.

Por que importa: permite depurar onde falhou e medir latência e custo por etapa.

Figura: exemplo ilustrativo do paper.

Seis perguntas de observabilidade

Pergunta	Sinal a observar
Concluiu a tarefa?	Status final da execução e, separadamente, verificação do resultado (testes, esquema, avaliador ou revisão humana)
Houve erros ou desvios?	Spans com erro, falhas de ferramenta, limites estourados, permissões negadas e guardrails acionados
Quando terminou e quanto levou?	Duração do run e de cada span; tokens e custo
O que fez em cada passo?	Trace hierárquico: ferramenta, argumentos e resultado
Quem pediu e o que foi autorizado?	Atributos de usuário final e eventos de decisão de permissão (auditoria e SIEM)
Houve loop ou desperdício?	Número de turnos e de chamadas por run

Concluiu? O trace não prova acerto

- **Status final:** diz como a execução terminou, não se o objetivo foi atingido.
- **Claude Agent SDK:** o resultado traz subtype, is_error, turnos, duração e custo (success, error_max_turns, error_during_execution; fonte de terceiros).
- **Falhas de conteúdo:** resposta errada dita com confiança muitas vezes não dispara status de erro.
- **OpenAI:** trace grading atribui notas a traces para achar erros em escala.
- **Desvios:** aparecem em eventos de permissão e guardrails; no Claude Code cada decisão é um evento com a origem.

● Importância

- Separar “terminou sem erro” (trace) de “resultado correto” (avaliação)

● Atenção

- Execução sem exceção prova apenas que o agente não quebrou

Registros de pacotes

Versão estável e data de publicação lidas por API no PyPI e no npm, em 10/10/2026.

Documentação oficial

Priorizada sempre que disponível, para cada fabricante e plataforma.

Fontes secundárias

Só onde faltou cobertura oficial; marcadas como “terceiros”. Comparativos de concorrentes, com cautela.

Marcação de validação na bibliografia

[A]

página aberta integralmente

[B]

trecho conferido contra a afirmação

[C]

dados obtidos por API de registro

- **Reatribuídas na validação:** afirmações passaram para fontes mais precisas (R17, R43 a R47); a avaliação do Logfire ficou ①.
- **Limitações:** SaaS sem versão única (a coluna mostra o SDK); preços excluídos; Weave, Opik e AgentOps não aprofundados; beta e GA refletem a data da consulta.

5 OpenTelemetry GenAI: o padrão aberto

- **Status Development:** as convenções GenAI são o candidato a linguagem comum, mas nomes ainda podem mudar.
- **12/06/2026:** na v1.42.0, foram movidas para o repositório semantic-conventions-genai.
- **Operações:** `create_agent`, `invoke_agent` e `execute_tool`; o span de agente é nomeado `invoke_agent` + nome do agente.
- **Métricas:** duração do run, número de chamadas de ferramenta e de inferência por run.
- **Quem já adota:** Datadog aceita OTel GenAI 1.37+; MLflow ingere e exporta; Google ADK implementa nativamente.

● **Implicação positiva (+)**

- Portabilidade entre fabricantes e backends

● **Implicação negativa (–)**

- Nomes podem mudar entre versões; o Claude emite spans próprios com prefixo `claude_code`.

Anthropic (Claude)

Versão: py 0.2.165 · npm 0.3.296 · CLI 2.1.296

Estado: Traces em beta; SDK 0.x

● Recursos

- Agent SDK: spans por interação, modelo e ferramenta, métricas e eventos; subagentes aninhados
- Contexto W3C propagado entre sua aplicação e o agente
- Managed Agents: tracing no Console e eventos de sessão, span e agente pela API

● Pontos fortes (+)

- OTLP aberto, sem backend obrigatório
- Conteúdo não é registrado por padrão, só a estrutura
- Eventos de permissão por usuário final alimentam um SIEM

● Fracos e limites (-)

- Traces em beta: nomes de spans podem mudar
- Falhas de exportação são silenciosas por padrão
- Spans de hook exigem configuração beta extra; Managed Agents exige header beta

● O que não faz

- O SDK não gera telemetria própria nem tem interface de análise
- Sem avaliadores nativos sobre traces nos documentos consultados

OpenAI (Agents SDK)

Versão: py 0.23.1 · JS 0.20.0

Estado: Tracing ativo por padrão; SDK 0.x

● Recursos

- Tracing ativo por padrão: run, tarefa, turno, agente, geração, ferramenta, guardrail, handoff e áudio
- group_id liga traces de uma mesma conversa
- Dashboard com trace grading e evals; cerca de 30 processadores externos

● Pontos fortes (+)

- Zero configuração e hierarquia rica em handoffs e guardrails
- Tracing gratuito mesmo com modelos de outros fornecedores

● Fracos e limites (-)

- Indisponível sob Zero Data Retention
- Entradas e saídas capturadas por padrão
- Redação em processador separado não barra o exportador padrão se falhar; exportação em lote

● O que não faz

- Formato nativo próprio: a ponte para OTel depende de instrumentadores de terceiros

Google (ADK e Agent Engine)

Versão: py 2.11.0 · JS 2.2.1

Estado: Major 2.x; telemetria nativa

● Recursos

- ADK implementa as convenções GenAI do OTel: agente como span raiz, filhos para LLM e ferramentas
- Contexto propagado entre processos
- Agent Engine exporta para o Cloud Trace; Python, Go e Kotlin

● Pontos fortes (+)

- OTel nativo
- Backend gerenciado com nível gratuito

● Fracos e limites (-)

- Cloud Logging descarta a entrada que excede o tamanho máximo de log

● O que não faz

- Cloud Trace é visualizador de traces
- Avaliação contínua e painéis específicos de agentes não apareceram nos documentos consultados

Microsoft (Foundry)

Versão: py 1.21.0 (azure-ai-projects 2.8.0)

Estado: GA: prompt e hospedados · preview: workflows e externos

● Recursos

- Tracing GA para agentes de prompt e hospedados; preview para workflows e externos
- Armazenamento no Application Insights
- Convenções OTel multiagente (com a Outshift, da Cisco); Agent Framework emite spans `invoke_agent`

● Pontos fortes (+)

- O mais avançado em semântica multiagente
- Integrado ao Azure Monitor

● Fracos e limites (–)

- Custo conforme volume e retenção do Application Insights
- Exige o papel Log Analytics Reader
- Status diverge entre páginas: o quickstart de agente hospedado ainda fala em preview

● O que não faz

- Fora do Azure, é preciso exportar via OTel para outro backend

6.5

AWS (Bedrock AgentCore)

Versão: py 1.24.1

Estado: Em evolução (log unificado em 23/07/2026)

● Recursos

- Sessão, trace e span; traces com ferramentas (entrada, saída e tempo) e erros
- Desde 23/07/2026: telemetria em um grupo de logs por agente, com IAM e CMK por agente
- Padrão em agentes criados desde 20/07/2026

● Pontos fortes (+)

- Dados ficam na conta do cliente, com a governança nativa da AWS
- Cobre também agentes hospedados fora do AgentCore

● Fracos e limites (-)

- Spans de agente exigem instrumentação com ADOT; por padrão só a memória gera spans
- Exige habilitar o CloudWatch Transaction Search

● O que não faz

- Sem avaliação de qualidade integrada nos documentos consultados

Versão: py 0.14.7 · JS 0.10.11

Estado: SaaS comercial; SDKs 0.x

● Recursos

- Runs equivalem a spans, em projetos; threads por metadata; assistente de análise (Polly)
- Ingere OTel; tracing distribuído, amostragem e prevenção de dados sensíveis
- Custo por token, dashboards e avaliação online

● Fracos e limites (–)

- Limite de runs por trace: excedentes são rejeitados
- Integração com o OpenAI Agents SDK é só Python

● Implicação positiva (+)

- Avaliação online e dashboards no mesmo produto
- Retenção de 400 dias no SaaS

● Licença e hospedagem

- SaaS com regiões na UE e na APAC
- Self-host restrito ao plano enterprise (segundo terceiros)

Versão: py 4.17.0 · JS @langfuse/tracing 5.13.1

Estado: Plataforma v4 (OTLP-first)

● Recursos

- Observações aninhadas, sessões, grafo de agentes e filas de anotação humana
- v4 prioriza ingestão por OTLP; roda sobre ClickHouse

● Implicação positiva (+)

- Núcleo MIT com self-host em qualquer plano
- Avaliação com anotação humana integrada

● Fracos e limites (–)

- Código aberto elimina a licença, não o custo de operar (terceiros)
- Relato de quebras na migração do SDK v3 para v4 (terceiros)

● Licença e hospedagem

- Opera em nuvem ou self-host, com núcleo sob licença MIT

Arize Phoenix e AX

Versão: Phoenix 20.20.0

Estado: Lançamentos frequentes; licença ELv2

● Recursos

- OTEL + OpenInference: tipos de span LLM, Tool, Agent e Retriever
- Evals, datasets, experimentos e gestão de prompts
- Instrumentadores para OpenAI Agents, LangChain e DSPy

● Implicação positiva (+)

- Pacote aberto com avaliação integrada
- AX acrescenta datastore proprietário e avaliações online (terceiros)

● Fracos e limites (–)

- Licença Elastic 2.0 restringe oferecer o software como serviço gerenciado (terceiros)
- Biblioteca de evals em TypeScript está em alpha

● Licença e hospedagem

- Phoenix: código disponível sob Elastic 2.0
- AX: serviço gerenciado

Datadog (Agent Observability)

Versão: ddtrace 4.15.6

Estado: SaaS

● Recursos

- Aceita OTel GenAI 1.37 ou superior
- Span links alimentam um Execution Graph de pipelines multiagente
- Auto-instrumentação para OpenAI Agents SDK, LangGraph e CrewAI

● Implicação positiva (+)

- Mostra como a saída de um span alimenta outro em pipelines multiagente
- Acompanha taxa de erro, latência, custo e decisões do agente

● Fracos e limites (–)

- Links para spans de outro trace são armazenados, mas não desenhados
- Produto não suportado nos sites gov

● Licença e hospedagem

- Oferecido apenas como SaaS

7.5

Braintrust

Versão: py 0.45.0 · JS 3.37.2

Estado: SaaS e self-host híbrido

● Recursos

- Avaliação offline e online sobre logs reais
- Integrações com Claude Agent SDK, OpenAI Agents e Google ADK
- Mascaramento de dados

● Implicação positiva (+)

- Avaliação e rastreamento no mesmo fluxo
- Integra com vários SDKs de agentes

● Fracos e limites (–)

- Self-host híbrido: a documentação descreve telemetria enviada ao plano de controle da empresa
- Custo cresce com o uso de scoring (terceiros)

● Licença e hospedagem

- Hospedagem: SaaS ou híbrida, com plano de dados no cliente

7.6

MLflow Tracing

Versão: 3.17.0

Estado: Código aberto

● Recursos

- Compatível com OTel: ingere e exporta convenções GenAI
- Mais de 30 integrações via autolog
- SDK enxuto para produção (mlflow-tracing)

● Implicação positiva (+)

- Portabilidade: OTel de ponta a ponta
- Uma chamada ativa o tracing de cada framework suportado

● Fracos e limites (–)

- É preciso chamar o autolog explicitamente para cada integração

● Licença e hospedagem

- Hospedagem: código aberto ou serviço gerenciado

7.7

Pydantic Logfire

Versão: 5.1.1

Estado: Produto comercial

● Recursos

- Camada sobre OTel com SDKs Python e JavaScript
- Integração com Pydantic AI e com um gateway de IA

● Implicação positiva (+)

- Qualquer aplicação com instrumentação OTel pode enviar dados

● Fracos e limites (–)

- Avaliações e gestão de prompts só em documentação espelhada, sem confirmação oficial (marca ①)
- Produto comercial

● Licença e hospedagem

- Hospedagem: SaaS; self-host apenas no plano Enterprise

Versões e maturidade (1/2)

Plataforma	Versão atual	Publicação	Estado de maturidade
Claude Agent SDK / Claude Code	py 0.2.165 · npm 0.3.296 · CLI 2.1.296	08–09/10/2026	Traces e telemetria em beta; numeração 0.x
Claude Managed Agents	API, sem pacote	—	Beta (header managed-agents-2026-04-01)
OpenAI Agents SDK	py 0.23.1 · JS 0.20.0	02/10 e 08/10/2026	Tracing ativo por padrão; numeração 0.x
Google ADK	py 2.11.0 · JS 2.2.1	02/10 e 06/10/2026	Major 2.x; telemetria nativa
Microsoft Agent Framework	py 1.21.0 (azure-ai-projects 2.8.0)	08/10 e 05/10/2026	GA para agentes de prompt e hospedados; preview para workflows e externos
AWS Bedrock AgentCore	py 1.24.1	07/10/2026	Em evolução (log unificado por agente em 23/07/2026)
LangSmith	py 0.14.7 · JS 0.10.11	09/10/2026	SaaS comercial; SDKs em 0.x

Versão = estável mais recente em 10/10/2026. Numeração 0.x indica SDK anterior à versão 1.0.

Versões e maturidade (2/2)

Plataforma	Versão atual	Publicação	Estado de maturidade
Langfuse	py 4.17.0 · JS @langfuse/tracing 5.13.1	05/10 e 07/10/2026	Plataforma v4 (OTLP-first)
Arize Phoenix	20.20.0	09/10/2026	Lançamentos frequentes; licença ELv2
Datadog	ddtrace 4.15.6	08/10/2026	SaaS
Braintrust	py 0.45.0 · JS 3.37.2	07/10 e 09/10/2026	SaaS e híbrido
MLflow Tracing	3.17.0	07/10/2026	Código aberto
Pydantic Logfire	5.1.1	25/09/2026	Produto comercial
OpenTelemetry (Python SDK)	1.45.1	06/10/2026	Convenções GenAI em Development

SaaS não tem versão única: a coluna mostra o SDK.

Matriz de recursos (1/2)

Plataforma	Passos (LLM e ferramenta)	Multiagente	OTel	Sessões e threads	Avaliação sobre traces	Hospedagem
Claude Agent SDK	✓ (beta)	✓ subagentes	✓ OTLP	✓ session.id	—	Backend à sua escolha
Claude Managed Agents	✓ (Console)	—	—	✓	—	Gerenciado
OpenAI Agents SDK	✓	✓ handoffs	🟡 via terceiros	✓ group_id	✓ trace grading	Dashboard OpenAI
Google ADK e Agent Engine	✓	🟡	✓ nativo	—	—	Cloud Trace
Microsoft Foundry e Agent Framework	✓	✓ semântica própria	✓	—	🟡	Application Insights
AWS AgentCore	✓ (com ADOT)	✓	✓ ADOT	✓	—	CloudWatch
LangSmith	✓	✓	✓	✓ threads	✓	SaaS; enterprise

✓ confirmado na documentação · 🟡 parcial ou condicionado · — não encontrado nos documentos consultados (não prova ausência)

Matriz de recursos (2/2)

Plataforma	Passos (LLM e ferramenta)	Multiagente	OTel	Sessões e threads	Avaliação sobre traces	Hospedagem
Langfuse	✓	✓ grafo	✓	✓	✓	SaaS; MIT
Phoenix e AX	✓	✓	✓	—	✓	ELv2; SaaS
Datadog	✓	✓ grafo	✓	—	● experimentos	SaaS
Braintrust	✓	✓	✓	—	✓ online e offline	SaaS; híbrido
MLflow	✓	✓	✓	—	—	OSS ou gerenciado
Logfire	✓	✓	✓	—	●	SaaS; enterprise

✓ confirmado na documentação · ● parcial ou condicionado · — não encontrado nos documentos consultados (não prova ausência)

● Convergência incompleta

Todos caminham para OTel, mas com dialetos: Claude (claude_code.*), OpenAI (formato próprio), Google e Microsoft (GenAI), AWS (ADOT).

● Maturidade desigual

Traces do Claude, tracing de workflows do Foundry e convenções GenAI estão em beta, preview ou Development: preveja migração.

● Privacidade e conteúdo

Claude Agent SDK não registra conteúdo por padrão; OpenAI Agents SDK registra entradas e saídas e não funciona sob ZDR.

● Falhas silenciosas

Em ao menos uma plataforma, falha de exportação não gera erro visível: trace ausente pode ser falha de exportação.

● Traço não é correção

Avaliação sobre traces: OpenAI, LangSmith, Langfuse, Phoenix e Braintrust (Logfire, parcial). As demais exigem integrar um avaliador.

Por que importa

Decisões de longo prazo devem prever migração de nomes de spans e atributos e definir, antes da produção, o que o trace pode conter.

9

Boas práticas

1

Instrumente por OTel sempre que possível: para manter a portabilidade entre fabricantes e backends.

2

Defina por escrito o que será capturado: conteúdo, argumentos e resultados, e quem pode ler.

3

Monitore a própria exportação de telemetria: com alerta para queda de volume de spans.

4

Separe duas métricas: “terminou sem erro” (sinal do trace) e “resultado correto” (sinal de avaliação).

5

Agrupe traces por sessão ou conversa: em tarefas longas ou com várias interações.

Estamos no meio do amadurecimento: soluções preliminares, várias em beta ou preview

- **Viável hoje:** rastrear cada passo de agentes funciona em todas as plataformas analisadas.
- **Beta:** traces do Claude Agent SDK e Managed Agents.
- **Preview:** workflows e agentes externos do Foundry.
- **Development:** convenções OTel GenAI. SDKs 0.x: Claude Agent SDK, OpenAI Agents SDK, LangSmith e Braintrust.
- **Custo de migração:** nomes de spans, atributos e comportamentos podem mudar entre versões.

QP1

O modelo de trace e span do tracing distribuído se aplica diretamente, com variações de nomenclatura.

QP2

Conclusão, erros, duração e passos saem do trace; a correção do resultado exige avaliação independente.

QP3

Ecossistema amplo, porém em transição: beta, preview e convenções ainda em Development.

Glossário (1/5)

● ADOT

AWS Distro for OpenTelemetry: distribuição do OTel da AWS para instrumentar agentes e enviar traces ao CloudWatch.

● Agente de IA

Sistema que usa um modelo de linguagem para decidir, em tempo de execução, quais ferramentas chamar e como concluir uma tarefa.

● Amostragem (sampling)

Registro de apenas uma fração dos traces, para reduzir volume e custo.

● Autolog

Ativação de instrumentação automática por uma única chamada (termo do MLflow).

● Avaliação (eval)

Medição da qualidade do resultado de um agente, com notas dadas por código, modelo avaliador ou pessoas.

● Backend de observabilidade

Sistema que recebe, armazena e permite consultar traces, métricas e logs.

● Beta, preview, GA, Development

Estágios de maturidade: beta e preview ainda mudam; GA é disponibilidade geral; Development é o estágio inicial das convenções OTel.

● CMK

Customer-managed key: chave de criptografia gerenciada pelo cliente.

● Convenções semânticas

Nomes e atributos padronizados para spans, métricas e eventos (ex.: `gen_ai.agent.name`).

● ELv2

Elastic License 2.0: código disponível, uso amplo, mas sem oferecer o software como serviço gerenciado.

● Evento

Registro estruturado de uma ocorrência (prompt, erro de API, decisão de permissão), distinto de métricas e spans.

● Falha semântica

Erro de conteúdo: o agente conclui sem exceção, mas entrega uma resposta errada.

● Guardrail

Verificação que valida ou bloqueia entradas ou saídas do agente.

● Handoff

Transferência de controle de um agente para outro.

● Hook

Código executado em um ponto definido do ciclo do agente, como antes do uso de uma ferramenta.

● Instrumentação

Código ou biblioteca que gera telemetria, de forma manual ou automática.

● Métrica

Medida numérica agregada ao longo do tempo (tokens, custo, latência).

● MIT

Licença permissiva de código aberto.

● Observabilidade

Capacidade de entender o estado interno de um sistema a partir da telemetria que ele emite.

● OpenInference

Camada de convenções sobre o OTel, criada pela Arize, com span kinds como LLM, Tool, Agent e Retriever.

● OpenTelemetry (OTel)

Padrão aberto para gerar e exportar telemetria: traces, métricas e logs.

● OTLP

Protocolo do OpenTelemetry para transporte de telemetria.

● Propagação de contexto

Passagem do ID do trace entre processos, para que spans de componentes distintos formem um só trace (W3C Trace Context).

● Self-host

Hospedagem do software na infraestrutura do próprio usuário.

● Sessão (ou thread)

Agrupamento de vários traces de uma mesma conversa ou interação.

● SIEM

Plataforma de gestão de eventos de segurança, usada para auditoria.

● Span

Trecho rastreado da execução: uma etapa com início, fim, pai, status e atributos.

● Span link

Ligação entre spans sem relação de pai e filho (ex.: a saída de um alimenta a entrada de outro).

● Subagente

Agente acionado por outro agente durante a execução.

● Trace

Registro completo de uma operação, composto por spans.

● Trace grading

Atribuição de notas a traces de agentes para achar erros e regressões (termo da OpenAI).

● Tracing distribuído

Técnica de registrar o caminho de uma requisição por vários componentes.

● Uso de ferramenta (tool use)

Chamada de um recurso externo pelo agente: API, banco de dados, busca, execução de código.

● ZDR

Zero Data Retention: o fornecedor não retém dados das requisições; no OpenAI Agents SDK, o tracing não funciona sob ZDR.

Bibliografia (1/7)

R1

Anthropic. Observability with OpenTelemetry (Agent SDK)<https://code.claude.com/docs/en/agent-sdk/observability>

[A]

R2

Anthropic. Monitoring (Claude Code)<https://code.claude.com/docs/en/monitoring-usage?m=1>

[A]

R3

Anthropic. Managed Agents: observability<https://platform.claude.com/docs/en/managed-agents/observability>

[B]

R4

OpenAI. Tracing (Agents SDK)<https://openai.github.io/openai-agents-python/tracing/>

[A]

R5

OpenAI. Trace grading<https://developers.openai.com/api/docs/guides/trace-grading>

[B]

R6

OpenAI. Evaluate agent workflows<https://developers.openai.com/api/docs/guides/agent-evals>

[B]

R7

Google. Agent activity traces (ADK)<https://adk.dev/observability/traces/>

[B]

Bibliografia (2/7)

R8

Google. Observability for agents (ADK)<https://adk.dev/observability/>

[B]

R9

Google Cloud. Trace an agent (Agent Engine)<https://cloud.google.com/agent-builder/agent-engine/manage/tracing>

[B]

R10

Google Cloud. Observability for AI agent developers<https://docs.cloud.google.com/stackdriver/docs/instrumentation/ai-agent-overview>

[B]

R11

Microsoft. Trace agent concept (Foundry)<https://learn.microsoft.com/en-us/azure/foundry/observability/concepts/trace-agent-concept>

[B]

R12

Microsoft. Set up tracing in Foundry<https://learn.microsoft.com/en-my/Azure/foundry/observability/how-to/trace-agent-setup>

[B]

R13

Microsoft. Quickstart: tracing a hosted agent<https://learn.microsoft.com/en-my/Azure/foundry/observability/quickstarts/quickstart-tracing-hosted-agent>

[B]

R14

Microsoft. Enable observability for agents (Agent Framework)<https://learn.microsoft.com/el-gr/agent-framework/tutorials/agents/enable-observability>

[B]

Bibliografia (3/7)

R15

AWS. Observability for agentic resources in AgentCore<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-telemetry.html>

[A]

R16

AWS. Add observability to AgentCore resources<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-configure.md>

[B]

R17

AWS. AgentCore: unified observability in a single log group (23/07/2026)<https://aws.amazon.com/about-aws/whats-new/2026/07/amazon-bedrock-agentcore-unified-observability-single-log-group/>

[B]

R18

OpenTelemetry. Semantic conventions for GenAI agent spans<https://opentelemetry.io/docs/specs/semconv/gen-ai/gen-ai-agent-spans>

[B]

R19

Dash0 (terceiros). OpenTelemetry GenAI semantic conventions explained<https://www.dash0.com/knowledge/opentelemetry-genai-semantic-conventions-explained>

[B]

R20

DEV Community (terceiros). OTel GenAI conventions are not stable yet<https://dev.to/azena-ai/opentelemetrys-genai-semantic-conventions-are-not-stable-yet-heres-what-actually-shipped-in-2026-3mke>

[B]

R21

LangChain. Observability concepts (LangSmith)<https://docs.langchain.com/langsmith/observability-concepts.md>

[B]

Bibliografia (4/7)

R22

LangChain. Trace with OpenTelemetry (LangSmith)<https://docs.langchain.com/langsmith/trace-with-opentelemetry/index.html>

[B]

R23

LangChain. Observability how-to guides (LangSmith)https://docs.smith.langchain.com/how_to_guides/tracing

[B]

R24

Langfuse. Engineering clarifications<https://langfuse.com/resources/engineering/clarifications.md>

[B]

R25

Langfuse. Data model<https://langfuse.com/docs/observability/data-model.md>

[B]

R26

Arize. Phoenix (repositório)<https://github.com/Arize-ai/phoenix>

[B]

R27

Atlan (terceiros). What is Arize<https://atlan.com/know/ai-agent/ai-agent-observability/what-is-arize/>

[B]

R28

Datadog. OpenTelemetry instrumentationhttps://docs.datadoghq.com/llm_observability/instrument/otel_instrumentation/

[B]

Bibliografia (5/7)

R29

Datadog. Agent monitoringhttps://docs.datadoghq.com/llm_observability/monitoring/agent_monitoring/

[B]

R30

Braintrust. Self-hosting<https://braintrust.dev/docs/guides/self-hosting>

[B]

R31

Braintrust. Release notes<https://www.braintrust.dev/docs/release-notes>

[B]

R32

Braintrust. Glossary<https://www.braintrust.dev/docs/reference/glossary.md>

[B]

R33

MLflow. Tracing for LLM and agent observability<https://mlflow.org/docs/latest/genai/tracing/index.html>

[B]

R34

Databricks. Automatic tracing and integrations<https://docs.databricks.com/gcp/en/mlflow3/genai/tracing/automatic-tracing>

[B]

R35

Pydantic. Logfire: AI observability<https://pydantic.dev/logfire.md>

[B]

Bibliografia (6/7)

R36

Inference.net (terceiros). Claude Agent SDK tracing and evaluation in production<https://inference.net/content/claude-agent-sdk-tracing/>

[B]

R37

ClickHouse (terceiros). What is AI agent observability<https://clickhouse.com/resources/engineering/what-is-ai-agent-observability>

[B]

R38

BenchLM (terceiros). Best LLM observability tools<https://benchlm.ai/blog/posts/best-llm-observability-tools>

[B]

R39

Jannik Reinhard (terceiros). Microsoft Foundry observability<https://jannikreinhard.com/microsoft-foundry-observability/>

[B]

R40

Registros PyPI e npm, consultados por API em 10/10/2026 (versões e datas)<https://pypi.org> · <https://registry.npmjs.org>

[C]

R41

Latitude (terceiros). LLM observability tools compared<https://latitude.so/blog/llm-observability-tools-compared>

[B]

R42

RFP.wiki (terceiros). Braintrust<https://www.rfp.wiki/vendors/braintrust>

[B]

Bibliografia (7/7)

R43

Pydantic. Logfire FAQ<https://logfire.pydantic.dev/docs/faq>

[B]

R44

Microsoft. Tracing in Microsoft Foundry (página anterior; menção à Outshift)<https://learn.microsoft.com/azure/ai-foundry/agents/concepts/tracing>

[B]

R45

Langfuse. Integração AI SDK C++ (modelo de dados v4, OTLP-first)<https://langfuse.com/integrations/frameworks/ai-sdk-cpp.md>

[B]

R46

Inference.net (terceiros). Arize Phoenix alternatives for agent observability<https://inference.net/content/arize-phoenix-alternatives/>

[B]

R47

Pydantic. Pydantic AI: Logfire<https://ai.pydantic.dev/logfire>

[B]