



Glossary - Agents

138 terms collected from the sheets in this collection; [brackets] name the source sheets

A

A2A (Agent2Agent protocol) — Open protocol, announced by Google in April 2025 and hosted by the Linux Foundation since June 2025, for agents built on different stacks to discover each other and exchange tasks. [*\[Multi-Agent Systems\]*](#)

Accessibility tree — Structured view of an interface (roles, names, states) that browser agents can act on instead of pixels. [*\[Computer and Browser Use; AI Agents: Self-Assessment\]*](#)

AG2 — Community fork of AutoGen (late 2024) that continues the original conversational API. [*\[Agent Frameworks and SDKs\]*](#)

Agent-as-tool — A sub-agent exposed to a parent agent as an ordinary tool: the parent calls it, it runs its own loop, it returns a result. [*\[Multi-Agent Systems\]*](#)

Agent Card — JSON document an A2A agent publishes (at a well-known URL) describing its skills, endpoint, input/output modes and authentication. [*\[Multi-Agent Systems\]*](#)

Agent loop — Model call, tool execution by the harness, result appended, repeated until a final answer or a budget stops it. [*\[AI Agents: Start Here; What Is an AI Agent? +1\]*](#)

Agent SDK — A library that implements the agent loop, tool dispatch and related plumbing so application code only declares agents, tools and policies. [*\[Agent Frameworks and SDKs\]*](#)

Agent (working definition) — A system where the LLM dynamically directs its own process and tool use, deciding the next step and when to stop (Anthropic, 'Building effective agents', Dec. 2024). [*\[What Is an AI Agent?\]*](#)

Agentic — A matter of degree, not a yes/no label: the more steps the model chooses, the more agentic the system. [*\[What Is an AI Agent?\]*](#)

Agentic search — Finding relevant code by letting the model call grep, glob and file reads step by step, instead of querying a pre-built embedding index. [*\[Coding Agents\]*](#)

AGENTS.md — An open, tool-neutral format for agent instructions in a repository, launched in 2025 and read by many coding agents. [*\[Coding Agents\]*](#)

AI agent — A system in which a language model decides, step by step, which actions (tool calls) to take and when to stop, inside a loop that feeds each result back to the model. [*\[AI Agents: Start Here\]*](#)

Artifact (A2A) — An output produced by a remote agent for a task (a document, data, a file), made of typed parts. [*\[Multi-Agent Systems\]*](#)

Augmented LLM — A model given retrieval, tools and memory; the building block of both workflows and agents. [*\[What Is an AI Agent?\]*](#)

AutoGen — Microsoft Research framework for conversational multi-agent systems, redesigned in v0.4 (2025) on an asynchronous actor model. [*\[Agent Frameworks and SDKs\]*](#)

Autonomy level — How much the system decides and acts without a human: from a single answer to unattended runs in production. [*\[What Is an AI Agent?\]*](#)

B

Blackboard — A shared store that all agents read and write (a file, a database, a task board), used instead of direct messages. [*\[Multi-Agent Systems\]*](#)

Browser agent — An agent that navigates websites for the user, through screenshots, the page's accessibility tree, DOM tools, or a mix. [*\[Computer and Browser Use\]*](#)

C

Checkpoint — A component that saves the agent's state after each step so a thread can be resumed, replayed or branched (LangGraph term). [*\[Agent Memory and State\]*](#)

Claude Agent SDK — Anthropic's library (Python, TypeScript) exposing the Claude Code harness: built-in file, shell and web tools, permissions, hooks, sub-agents, MCP, compaction; renamed from Claude Code SDK in 2025. [*\[Agent Frameworks and SDKs\]*](#)

Code-as-action — Agent design where the model writes executable code instead of single JSON tool calls (smolagents' CodeAgent). [*\[Agent Frameworks and SDKs; AI Agents: Self-Assessment\]*](#)

Codebase index — Embeddings of a repository's files or chunks, queried by similarity to fetch code relevant to a request. [*\[Coding Agents\]*](#)

Coding agent — An agent whose tools read, edit and run code in a repository (Claude Code, Codex, Cursor agent, and others). [*\[AI Agents: Start Here; Coding Agents\]*](#)

Compaction — Summarising a long history so an agent continues in a smaller context. [*\[Agent Memory and State; AI Agents: Self-Assessment\]*](#)

Compounding error — Failure probability that grows with the number of steps: n steps that each succeed with probability p all succeed with probability p to the power n. [*\[What Is an AI Agent?\]*](#)

Computer use — An agent capability where the model sees screenshots of a desktop and returns mouse and keyboard actions that the harness performs. [*\[Computer and Browser Use\]*](#)

Computer-use agent — An agent that operates a graphical interface from screenshots (or an accessibility tree) with mouse and keyboard actions. [*\[AI Agents: Start Here\]*](#)

Constrained decoding — Generation that masks every token which would break the target grammar or schema, so the output always parses. [*\[Tool Use and Function Calling\]*](#)

Context isolation — Giving a sub-agent a fresh context window so its exploration does not fill the parent's; only a summary comes back. [*\[Multi-Agent Systems\]*](#)

Context rot — The drop in a model's accuracy as its input grows longer, even when the needed information is present. [*\[Agent Memory and State\]*](#)

Context window — The maximum number of tokens a model can read in one call: instructions, tools, history and tool results all share it. [*\[AI Agents: Start Here\]*](#)

CrewAI — Python framework for role-based multi-agent 'crews' (agents with role, goal, backstory; tasks; processes) plus Flows for deterministic orchestration. [*\[Agent Frameworks and SDKs\]*](#)

D

DOM (Document Object Model) — The browser's in-memory tree of a web page's HTML elements, which scripts and automation tools query and act on with selectors. [*\[Computer and Browser Use\]*](#)

Durable execution — Running an agent so that its state survives crashes and restarts, resuming from the last checkpoint instead of starting over. [*\[Agent Frameworks and SDKs\]*](#)

E

Edit tool — The tool a coding agent uses to change files: exact string replacement, a patch format, or whole-file writes. [*\[Coding Agents\]*](#)

Elicitation — A client feature (spec 2025-06-18) that lets a server ask the user for structured input or send them to a URL during a call. [*\[Model Context Protocol \(MCP\)\]*](#)

Environment — Everything outside the model that the agent observes and changes: files, APIs, a browser, a database, a shell. [*\[What Is an AI Agent?\]*](#)

Episodic memory — Records of specific past events or runs: what was tried, what happened, when. [*\[Agent Memory and State\]*](#)

Evaluator-optimizer — Generate, evaluate against criteria, revise; repeated until the evaluation passes. [*\[Planning and Reasoning Patterns for Agents; AI Agents: Self-Assessment\]*](#)

Exponential backoff — Retrying after waits that double each attempt (with random jitter) to ride out rate limits and transient failures. [*\[Tool Use and Function Calling\]*](#)

F

FAIL_TO_PASS / PASS_TO_PASS — SWE-bench test sets: tests that must go from failing to passing with the fix, and tests that must keep passing. [*\[Coding Agents\]*](#)

Function calling — The model's ability to answer with a structured request to call a named function with JSON arguments instead of (or before) plain text; also called tool use. [*\[Tool Use and Function Calling\]*](#)

G

Git worktree — An extra working directory attached to the same repository on another branch, letting several agents work in parallel without touching each other's files. [*\[Coding Agents\]*](#)

Golden set — A curated collection of representative tasks with their expected outcomes or checks, run on every change as a regression suite. [*\[Agent Evaluation and Reliability\]*](#)

Grounding (GUI) — Mapping an intended action ('click Submit') to the exact element or pixel coordinates on screen. [*\[Computer and Browser Use\]*](#)

Guardrail — A check outside the model that blocks, rewrites or escalates an input, a tool call or an output that breaks a rule. [*\[Agent Evaluation and Reliability\]*](#)

H

Handoff — Transfer of control and conversation from one agent to another, which then answers the user. [*\[Multi-Agent Systems; AI Agents: Self-Assessment\]*](#)

Harness — The program around the model that runs the loop: tool dispatch, permissions, limits, context management, logging. [*\[AI Agents: Start Here; What Is an AI Agent?\]*](#)

Headless mode — Running a coding agent non-interactively from a script or CI job with a prompt and a fixed tool allowlist. [*\[Coding Agents\]*](#)

Hook (agent) — A script the harness runs automatically at a lifecycle event (before or after a tool call, at session end) to block, check or log an action. [*\[Coding Agents\]*](#)

Human-in-the-loop — A design where a person approves, corrects or takes over specific agent actions, usually the irreversible or costly ones. [*\[AI Agents: Start Here\]*](#)

Human-in-the-loop (HITL) — Points in a run where a person approves, edits or rejects an action, or takes over the case. [*\[Agent Evaluation and Reliability\]*](#)

I

Idempotency key — A unique id sent with a write request so that a retried request is applied only once. [*\[Tool Use and Function Calling\]*](#)

Indirect prompt injection — Instructions hidden in content the agent reads (a web page, an e-mail) that try to redirect it; mechanics in responsible-ai. [*\[Computer and Browser Use\]*](#)

Intrinsic self-correction — Critique with no outside feedback; studies (Huang et al., 2023) found it often fails to improve reasoning and can make it worse. [*\[Planning and Reasoning Patterns for Agents\]*](#)

J

JSON-RPC 2.0 — A small remote-procedure-call format: requests with id, method and params; responses with the same id and a result or error; notifications without id. [*\[Model Context Protocol \(MCP\)\]*](#)

JSON Schema — A standard vocabulary (type, properties, required, enum, pattern...) that describes the shape of a JSON document and lets code validate it. [*\[Tool Use and Function Calling\]*](#)

L

LangGraph — LangChain's library for agents as explicit state graphs (nodes, edges, shared state) with checkpointing, interrupts for human approval and durable execution. [*\[Agent Frameworks and SDKs\]*](#)

LlamaIndex — Framework centred on connecting LLMs to data (loaders, indexes, query engines) with event-driven Workflows and agent classes on top. [*\[Agent Frameworks and SDKs\]*](#)

LLM-as-judge — Using a model with a written rubric to grade outputs or trajectories that code cannot check; must itself be validated against human labels. [*\[Agent Evaluation and Reliability\]*](#)

Lost in the middle — Finding (Liu et al., 2023) that models use information at the start and end of a long context better than in the middle. [*\[Agent Memory and State\]*](#)

M

MCP client — The connector inside a host that holds one stateful session with one MCP server and speaks the protocol on the host's behalf. [*\[Model Context Protocol \(MCP\)\]*](#)

MCP host — The AI application the user works in (Claude Code, Claude Desktop, an IDE, ChatGPT); it owns the model and creates one MCP client per server. [*\[Model Context Protocol \(MCP\)\]*](#)

MCP (Model Context Protocol) — An open protocol, introduced by Anthropic in November 2024, that lets any AI application connect to tool, data and prompt servers in one standard way. [*\[AI Agents: Start Here\]*](#)

MCP server — A program that exposes tools, resources and prompts over MCP, locally as a subprocess or remotely over HTTP. [*\[Model Context Protocol \(MCP\)\]*](#)

Memory consolidation — Merging, updating and deleting stored memories so they stay consistent: a new address replaces the old one. [*\[Agent Memory and State\]*](#)

Memory poisoning — Getting a false fact or a malicious instruction written into an agent's long-term memory so it persists across sessions. [*\[Agent Memory and State\]*](#)

Memory store — External storage for long-term memory: vector database, key-value profile, knowledge graph, plain files. [*\[Agent Memory and State\]*](#)

Multi-agent system — Several LLM agents, each with its own instructions, tools and context, that cooperate on one task by passing messages or control. [*\[Multi-Agent Systems\]*](#)

O

Observation — The result of an action fed back to the model: tool output, error message, screenshot, test log. [*\[What Is an AI Agent?\]*](#)

OpenAI Agents SDK — OpenAI's lightweight framework (2025, Python and TypeScript) built on agents, tools, handoffs, guardrails, sessions and built-in tracing; successor of the experimental Swarm. [*\[Agent Frameworks and SDKs\]*](#)

Orchestrator-workers — A central LLM breaks a task down at run time, delegates sub-tasks to worker calls and synthesises their results. [*\[Planning and Reasoning Patterns for Agents\]*](#)

OSWorld — Benchmark of 369 real desktop tasks in virtual machines (Ubuntu, Windows apps) checked by execution-based scripts (Xie et al., 2024). [*\[Computer and Browser Use\]*](#)

P

Parallel tool calls — Several tool calls emitted in one model turn, which the harness may execute concurrently and must answer together. [*\[Tool Use and Function Calling\]*](#)

Parallelisation (sectioning) — Running independent sub-tasks as simultaneous LLM calls and combining the outputs in code. [*\[Planning and Reasoning Patterns for Agents\]*](#)

pass^k — Probability that all k independent attempts succeed, p to the power k; the consistency users experience. [*\[Agent Evaluation and Reliability; AI Agents: Self-Assessment\]*](#)

Permission mode — The setting that decides which agent actions run without asking: read-only planning, ask on each action, auto-accept edits, or no prompts. [*\[Coding Agents\]*](#)

Plan-and-execute — The agent first writes an explicit multi-step plan, then executes steps one by one, re-planning when

an observation invalidates the plan. [*\[Planning and Reasoning Patterns for Agents\]*](#)

Playwright — Open-source browser automation library (Microsoft) used by many browser agents and by the Playwright MCP server. [*\[Computer and Browser Use\]*](#)

Procedural memory — Knowledge of how to do things: system prompts, rules files, skills, tool code; it shapes behaviour rather than being looked up. [*\[Agent Memory and State\]*](#)

Prompt caching — Reusing the processed prefix of a prompt (system, tools, earlier turns) across calls, billed at a fraction of the normal input price. [*\[What Is an AI Agent?\]*](#)

Prompt chaining — A workflow that splits a task into fixed sequential LLM calls, each consuming the previous output, with optional programmatic checks (gates) between them. [*\[Planning and Reasoning Patterns for Agents\]*](#)

Prompt (MCP) — A parameterised message template offered by a server and picked by the user, often surfaced as a slash command. [*\[Model Context Protocol \(MCP\)\]*](#)

Protected Resource Metadata — RFC 9728 document an MCP server publishes so clients discover which authorization server issues its tokens. [*\[Model Context Protocol \(MCP\)\]*](#)

Pydantic AI — Agent framework from the Pydantic team focused on type safety: typed dependencies, Pydantic-validated outputs, model-agnostic providers. [*\[Agent Frameworks and SDKs\]*](#)

R

Re-planning — Rewriting the remaining plan after a step fails or reveals new facts, instead of blindly continuing the original plan. [*\[Planning and Reasoning Patterns for Agents\]*](#)

ReAct — Reason + Act (Yao et al., 2022): the model alternates a reasoning trace (Thought), a tool call (Action) and the tool's output (Observation). [*\[What Is an AI Agent?\]*](#)

Reflection — The model critiques its own output or trajectory in words and uses the critique in the next attempt (Self-Refine, Reflexion). [*\[Planning and Reasoning Patterns for Agents\]*](#)

Reflexion — After a failed episode, the agent writes a verbal lesson to memory and retries with it (Shinn et al., 2023); works best with an external signal such as unit tests. [*\[Planning and Reasoning Patterns for Agents\]*](#)

Resource (MCP) — Read-only context identified by a URI (a file, a table schema, a ticket) that the application chooses

to attach to the model's context. [*\[Model Context Protocol \(MCP\)\]*](#)

Roots — Filesystem or URI boundaries the client tells a server it may work within, such as the open project folder. [*\[Model Context Protocol \(MCP\)\]*](#)

Routing — Classifying an input first, then sending it to a specialised prompt, tool set or model (for example a cheap model for easy cases). [*\[Planning and Reasoning Patterns for Agents\]*](#)

Rules file — A Markdown file at the repository root (CLAUDE.md, AGENTS.md) that the agent loads at the start of every session: commands, conventions, prohibitions. [*\[Coding Agents\]*](#)

S

Sampling (MCP) — A client feature that lets a server ask the host's model for a completion, under the host's control and the user's approval. [*\[Model Context Protocol \(MCP\)\]*](#)

Sandbox — An isolated environment (OS-level sandbox, container or cloud VM) that limits what the agent's commands can read, write and reach on the network. [*\[Coding Agents\]*](#)

Screenshot-action loop — Repeat: capture the screen, send it to the model, receive one action (click, type, scroll, key), execute it, capture again. [*\[Computer and Browser Use\]*](#)

Self-assessment — A set of questions answered without notes, then checked against the answers and reasons to find the sheets to re-read. [*\[AI Agents: Self-Assessment\]*](#)

Self-consistency — Sampling several reasoning paths at non-zero temperature and returning the most frequent final answer. [*\[Planning and Reasoning Patterns for Agents\]*](#)

Self-Refine — Same model generates, critiques and refines its output in a loop (Madaan et al., 2023). [*\[Planning and Reasoning Patterns for Agents\]*](#)

Semantic error — Wrong arguments, not found, permission denied: return it as a tool result with `is_error` so the model can change its plan. [*\[Tool Use and Function Calling\]*](#)

Semantic memory — General facts the agent has learned: user preferences, project conventions, domain knowledge. [*\[Agent Memory and State\]*](#)

Set-of-Marks — Prompting technique that overlays numbered boxes on interactive elements of a screenshot so the model answers with a mark number instead of pixel coordinates (Yang et al., 2023). [*\[Computer and Browser Use\]*](#)

smolagents — Hugging Face’s minimal agent library whose CodeAgent writes its actions as Python code executed in a sandbox. [*\[Agent Frameworks and SDKs\]*](#)

Span — One timed operation inside a trace, such as a single model call or tool execution, with its attributes. [*\[Agent Evaluation and Reliability\]*](#)

stdio transport — The client starts the server as a subprocess and exchanges newline-delimited JSON-RPC messages over its standard input and output. [*\[Model Context Protocol \(MCP\)\]*](#)

Step budget — The maximum number of loop iterations (or tokens, or seconds) a run may use before it is stopped. [*\[What Is an AI Agent?\]*](#)

Stop reason — The field of a model response saying why generation ended: final answer, tool call requested, token limit reached, stop sequence. [*\[What Is an AI Agent?\]*](#)

Streamable HTTP — The remote MCP transport: one HTTP endpoint, POST for client messages, optional Server-Sent Events for streaming. [*\[Model Context Protocol \(MCP\); AI Agents: Self-Assessment\]*](#)

Structured output — A model response guaranteed or validated to match a given JSON Schema, by constrained decoding, a forced tool or validate-and-retry. [*\[Tool Use and Function Calling\]*](#)

Sub-agent — An agent started by another agent with its own fresh context to do one bounded task and report back a summary. [*\[AI Agents: Start Here\]*](#)

Supervisor topology — One coordinating agent decides which worker agent acts next and gathers the results; workers do not talk to each other. [*\[Multi-Agent Systems\]*](#)

Swarm topology — Decentralised design where agents hand control to each other directly; whichever agent is active answers the user. [*\[Multi-Agent Systems\]*](#)

SWE-bench — Benchmark (Jimenez et al., 2023) of real GitHub issues from Python repositories; an agent’s patch counts as resolved when the issue’s hidden tests pass and existing tests still pass. [*\[Coding Agents\]*](#)

SWE-bench Verified — 500 human-validated GitHub issues used to measure how often a coding agent’s patch passes the hidden tests. [*\[AI Agents: Self-Assessment\]*](#)

T

Takeover — Handing control from the agent to the human for a step the agent must not or cannot do: passwords, CAPTCHAs, payments. [*\[Computer and Browser Use\]*](#)

Task (A2A) — The unit of work between A2A agents, with an id and a lifecycle state such as submitted, working, input-required, completed, failed or canceled. [*\[Multi-Agent Systems\]*](#)

Task success rate — Share of evaluation tasks whose final state or answer passes an automatic or human check. [*\[Agent Evaluation and Reliability\]*](#)

Time horizon (METR) — The length of tasks, measured in human working time, that an agent completes with 50 % success. [*\[Agent Evaluation and Reliability\]*](#)

Token multiplier — How many times more tokens a design spends than a plain chat on the same request; multi-agent designs have high multipliers. [*\[Multi-Agent Systems\]*](#)

Tool — A function the application exposes to the model by name, description and input schema; the model asks for it, the application runs it. [*\[AI Agents: Start Here\]*](#)

Tool annotation — Metadata on a tool such as read-only, destructive or idempotent, used by the harness to decide what needs approval. [*\[Tool Use and Function Calling\]*](#)

tool_choice — API parameter letting the model decide, forcing a tool call, forcing one named tool, or forbidding tools. [*\[Tool Use and Function Calling; AI Agents: Self-Assessment\]*](#)

Tool definition — Name, natural-language description and JSON Schema of the inputs that the application sends to the model with each request. [*\[Tool Use and Function Calling\]*](#)

Tool poisoning — Hiding instructions for the model in a tool’s description or results so that connecting the server hijacks the agent. [*\[Model Context Protocol \(MCP\)\]*](#)

Tool result — The message that returns a tool’s output (or error) to the model, linked to the call by its id. [*\[Tool Use and Function Calling\]*](#)

Tool-result clearing — Removing or stubbing old tool outputs from the history once they are no longer needed, the cheapest form of context editing. [*\[Agent Memory and State\]*](#)

Tool search — Loading only the tool definitions relevant to the current task (searched from a large catalogue) instead of sending all of them on every call. [*\[Tool Use and Function Calling\]*](#)

Trace — The tree of timed spans (model calls, tool calls, retrievals) recorded for one agent run, with inputs, outputs, tokens and errors. [*\[Agent Evaluation and Reliability\]*](#)

Trajectory — The full ordered record of an agent run: every model output, tool call, tool result and the final answer. [*\[AI Agents: Start Here\]*](#)

Trajectory evaluation — Grading the path of a run (tool choices, arguments, order, wasted steps, policy breaches), not only its final result. [*\[Agent Evaluation and Reliability\]*](#)

Transient error — Timeouts, 429, 503: retry in code with back-off, the model never needs to see them unless retries are exhausted. [*\[Tool Use and Function Calling\]*](#)

Tree of thoughts — Search over partial solutions (thoughts): generate several candidates per step, score them with the model, expand the best, backtrack from dead ends (Yao et al., 2023). [*\[Planning and Reasoning Patterns for Agents\]*](#)

Tripwire — A guardrail result that immediately halts the run (OpenAI Agents SDK term), for example when a jailbreak classifier fires. [*\[Agent Evaluation and Reliability\]*](#)

V

Verifier — Anything that checks a result independently of the generator: unit tests, a compiler, a schema, a rule engine, a second model, a human. [*\[Planning and Reasoning Patterns for Agents\]*](#)

Voting — Running the same task several times (or with several prompts) and aggregating the answers, for example by majority. [*\[Planning and Reasoning Patterns for Agents\]*](#)

W

WebArena — Benchmark of 812 web tasks on self-hosted copies of realistic sites (shop, forum, code hosting, maps, CMS) (Zhou et al., 2023). [*\[Computer and Browser Use\]*](#)

Workflow (LLM) — LLM calls and tools orchestrated through predefined code paths; the developer, not the model, decides the sequence. [*\[AI Agents: Start Here; What Is an AI Agent?\]*](#)

Working memory (agent) — Everything inside the current context window: instructions, tool definitions, conversation, tool results; lost when the call ends unless written elsewhere. [*\[Agent Memory and State\]*](#)