

Agentic mindset for startups

YREB - [Young Researcher Entrepreneurship Bootcamp](#)

Nicki Skafte Detlefsen, Associate Professor, DTU Compute

28/05-2026

Agenda

★ Lecture: What the heck is an Agents? and vibe coding intro(25min)

★ Exercise 1: Agents for buisness model canvas (25min)

★ Break (10min)

★ Exercise 2: Long running tasks (10min)

★ Lecture: Agentic engineering (30min)

★ Exercise 3: Build an MVP (80min)

– lunch break (60min) –

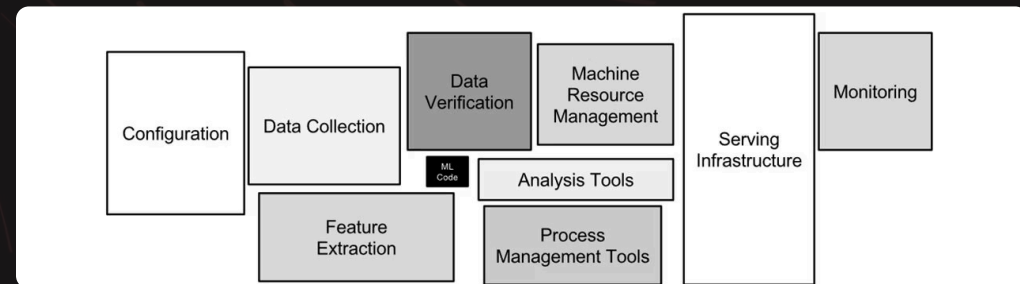
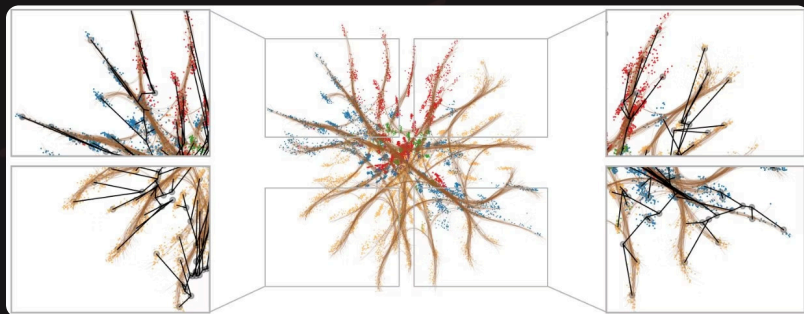
★ Presentations of MVPs (40min)

★ Wrap out (20min)

As usual i have prepared way to much
so lets see how far we get.

Who am I?

- 💡 Associate Professor at DTU Compute
 - 💡 Senior Researcher at CAISA
 - 💡 Research in MLOps, efficient machine learning, machine learning development, Agentic Engineering
 - 💡 Open-source developer
 - 💡 Part-time ML Engineer at <https://lightning.ai/>
 - 💡 Technical advisor for 5 DTU startups
- = A good mix between the academic and industrial world



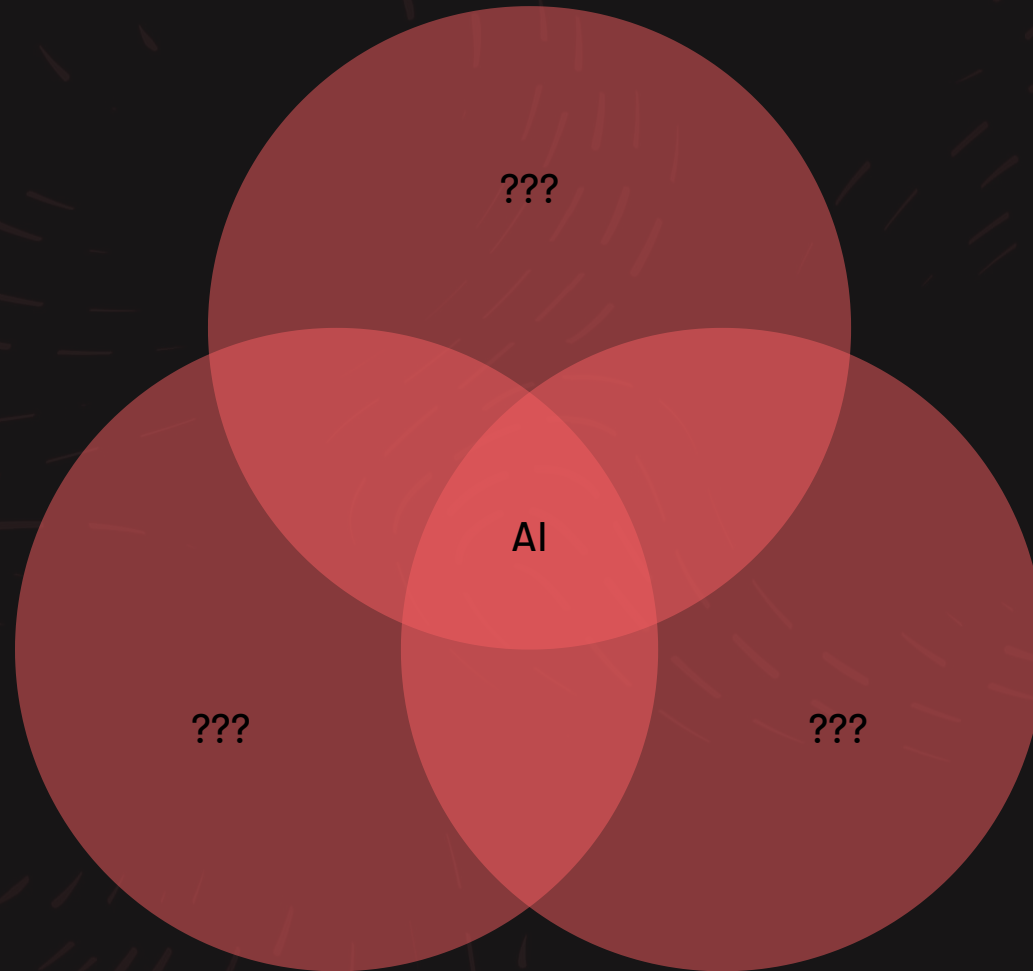
Preface to this workshop

Anyone who says they are an agentic engineering expert is full of bullshit

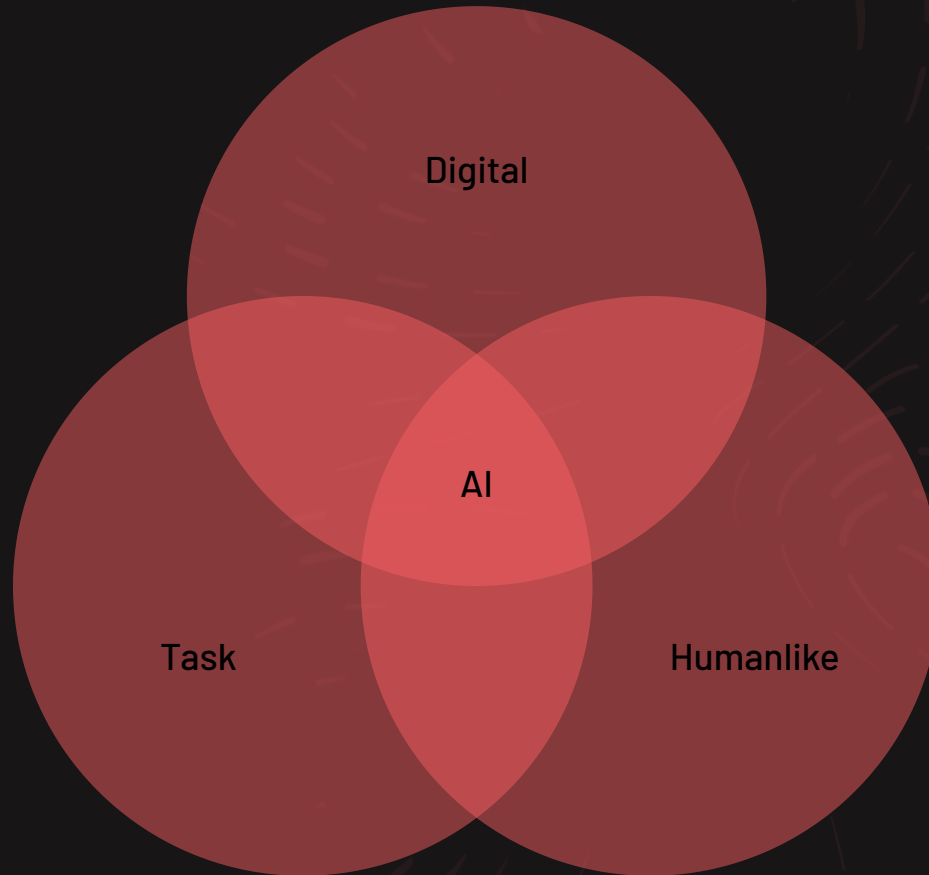
Nicki Skafte Detlefsen, expert in agentic engineering

We are in the fuck around and find out phase... so take everything presented as common knowledge with a grain of salt

What is useful AI? (Exercise)



What is useful AI?



Digital + Task, but not Humanlike

- Sorting algorithm, Thermostat controller, Barcode scanner app

Digital + Humanlike but no task:

- Deepfake video, Virtual influencer avatar

Task + Humanlike but not digital:

- Skilled craftsman, Stage actor following a script, Trained service animal

things can *look* like AI without *doing* anything

What is an AI Agent?

An AI Agent is an **autonomous entity** (to some level) that observes its environment through sensors, processes that information to make decisions, and then acts upon its environment through effectors to achieve its goals.



Perceive

Gathers data and observes the environment.



Deliberate

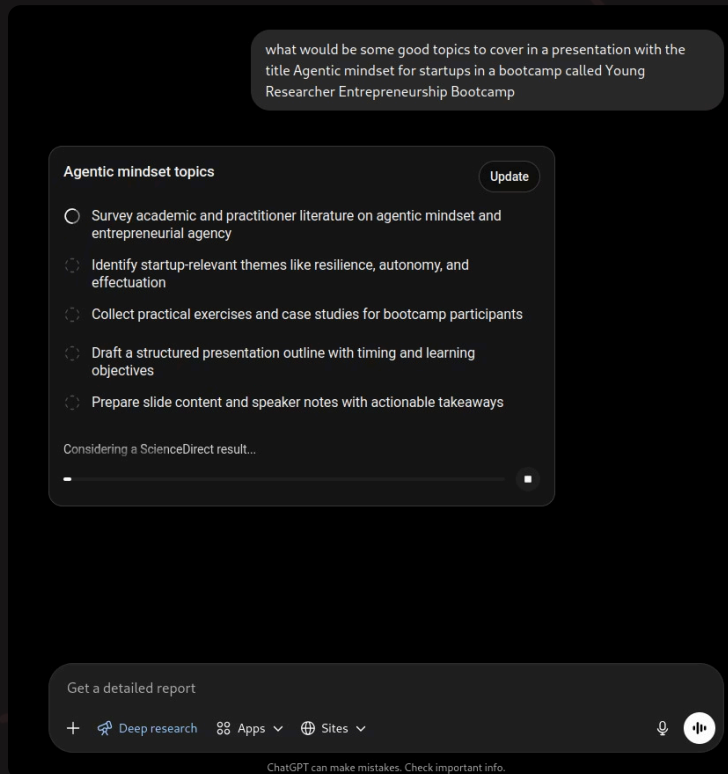
Processes information, plans, and makes decisions.



Act

Executes chosen actions to achieve specific goals.

An example: Deep research AI agent



✓ Digital - internet search

✓ Task - Produce a research report

✓ Humanlike - Requires effort to condense and summaries over many sources

✓ Autonomously - Self determines when it done searching the web

[1] <https://openai.com/index/introducing-deep-research/>

Why does this matter for startups

Startups win asymmetrically with speed and focus, and AI agents let small teams deploy capability instead of hiring it.



Deploy, don't hire

A solo founder can deploy agents for support, research, and coding at the same time.



Spin up in an afternoon

Work that once needed a hire or agency can now be set up fast.

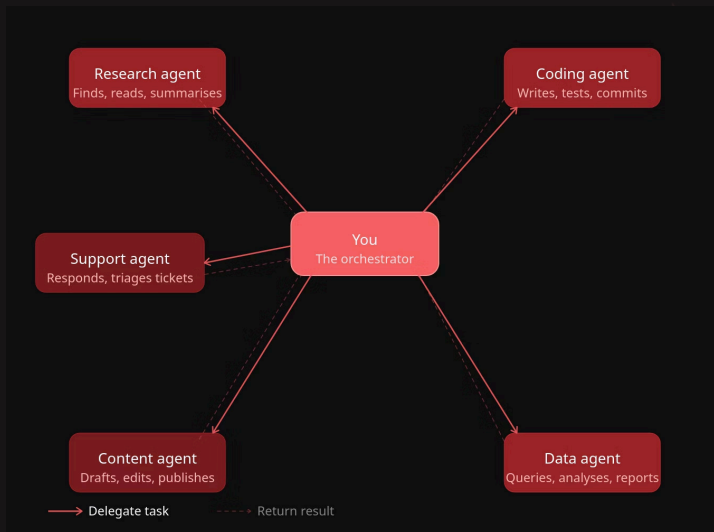


Judgment is the new constraint

The scarce resource shifts from manpower to deciding what work matters.

okay, but how do you actually think and work that way?

Agentic mindset



The **agentic mindset** is the habit of decomposing goals into tasks, and asking which tasks can be delegated to an autonomous agent – rather than asking how to do the task yourself.

"How do I do this?"

transforms into

"What needs to happen, and who or what should do it?"

Goal-first thinking

You specify the outcome, not the steps.

Trust calibration

Knowing when to let an agent run autonomously vs. when to keep a human in the loop.

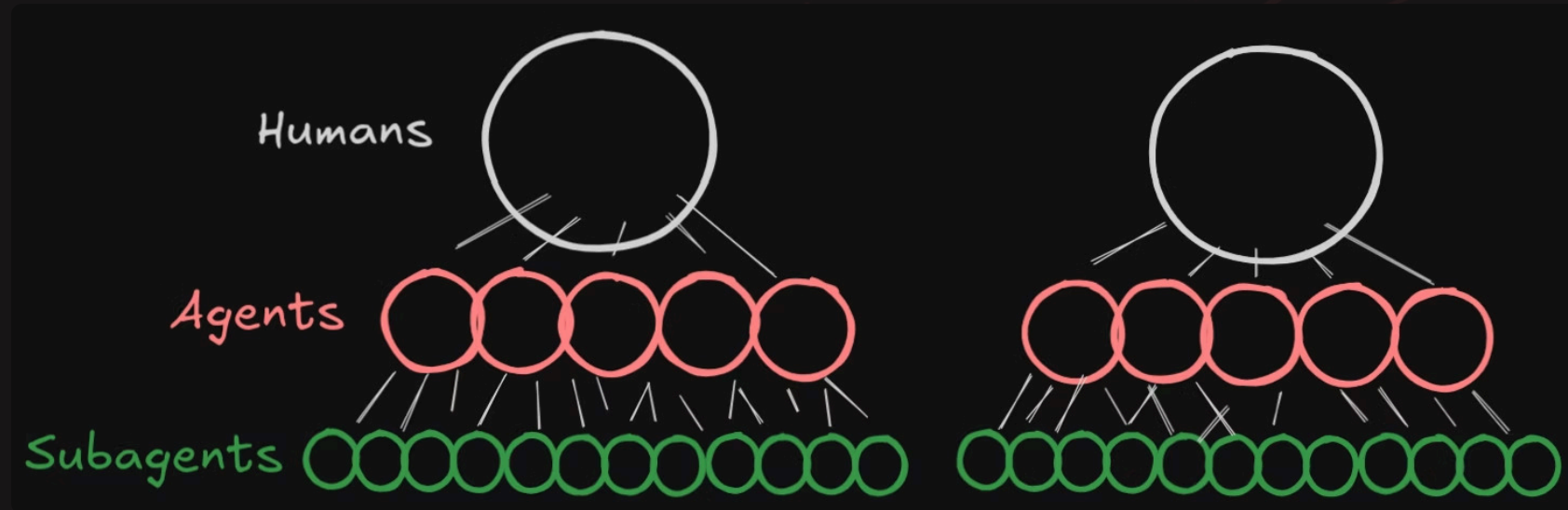
Task decomposition

Breaking ambiguous goals into concrete, delegatable subtasks.

Tolerance for imperfect autonomy

Accepting that an agent doing 80% of a task well is often more valuable than doing 100% yourself.

Agentic Mindset: Team vs. Individual



An agentic mindset means shifting from individual task execution to thinking like a team. Processes must be designed for **scalability from day one**.

Imagine a small team of 2-5 people, each leveraging 5-10 AI agents. This quickly scales their collective capability, allowing them to operate with the output of a much larger, **medium-sized company**. Therefore, the core of the agentic mindset revolves around mastering **parallel workstreams**.

In coding, tools like `git worktrees` offer a tangible example of enabling parallel development.

Don't just use agents – build them

Tools like Manus, Perplexity, and Agentforce are powerful. But they are someone else's vision of what an agent should do. As a startup, that's a ceiling.



Off-the-shelf agents

Optimized for the average use case – not your customers, your data, your workflow.



The real leverage

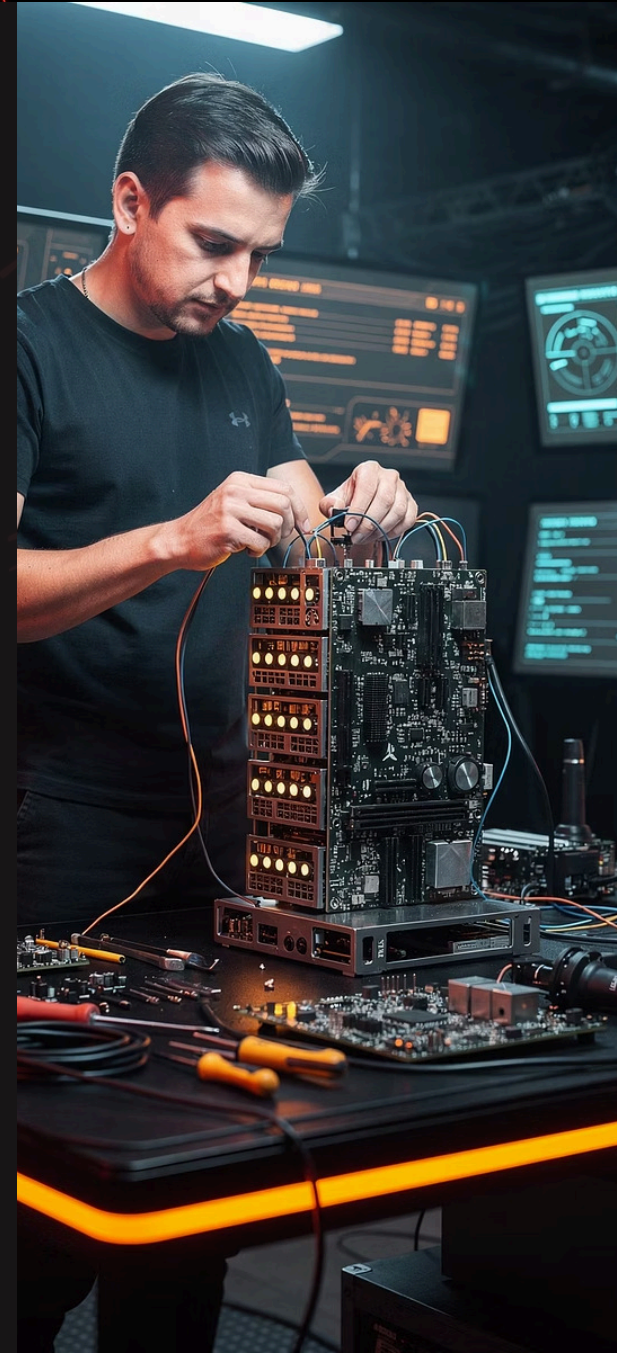
Composing your own agents around your unfair advantages: domain knowledge, proprietary data, unique process.



The barrier is lower than you think

The same building blocks power everything from Claude Code to enterprise automation pipelines.

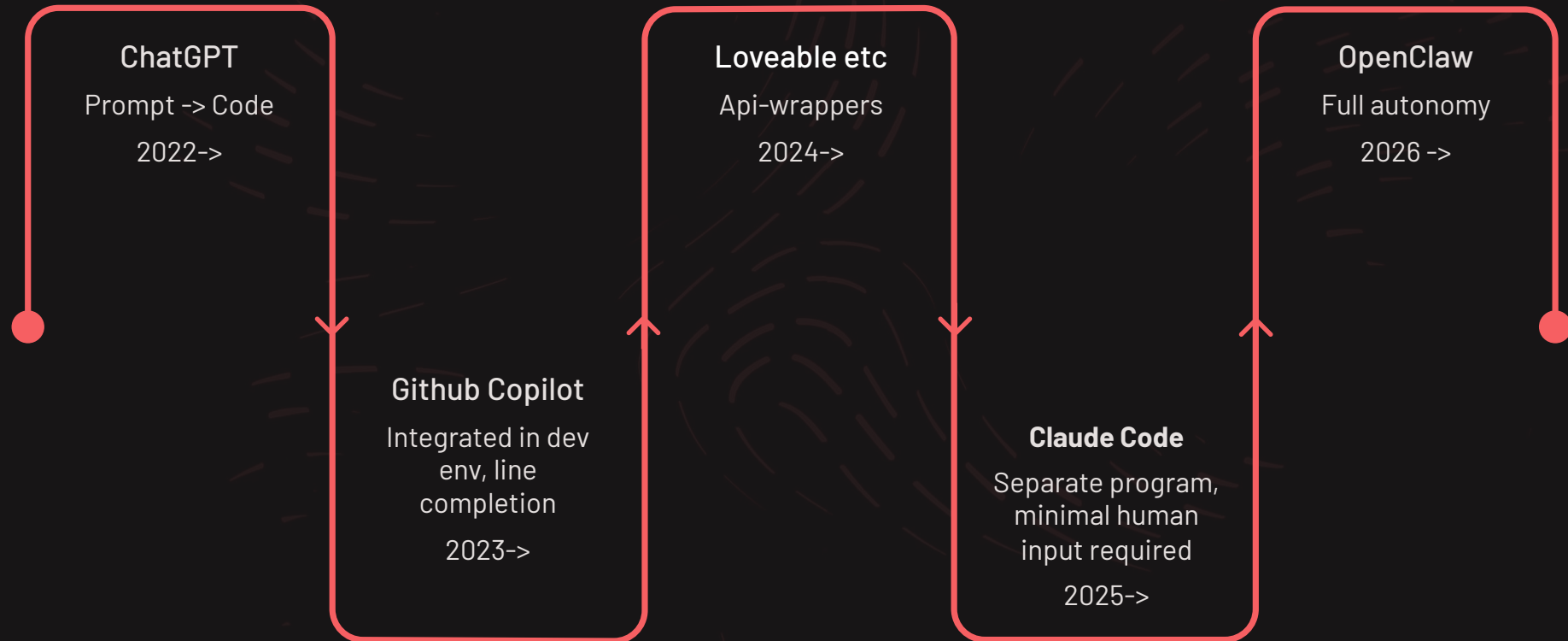
The startups that win won't be the ones who used the best tools. They'll be the ones who understood the tools well enough to build better ones.



Lets figure out how to build and build with Agents

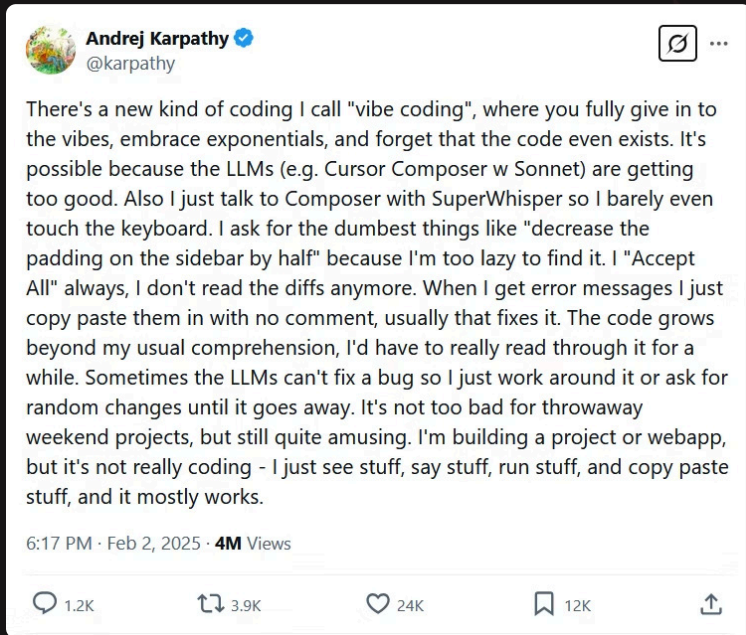
Getting started with vibe coding

How did we get here



In this workshop we will be working with level 3 - Claude Code

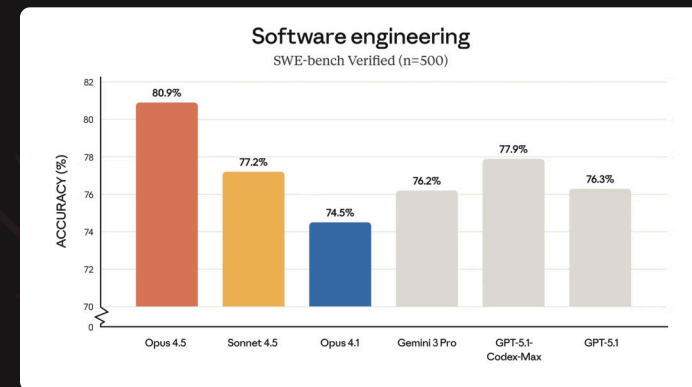
What is Vibe Coding?



Andrej Karpathy - OpenAI co-founder - AI chief at Tesla

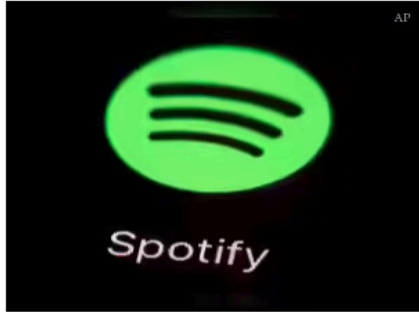
The term refers to a coding method that relies on LLMs, making it possible for programmers to generate working code by **providing natural language descriptions** instead of manually writing in a formal programming language.

Note that vibe coding has generally only really worked since Opus 4.5 (December 2025)



[1] <https://www.anthropic.com/news/claude-opus-4-5>

Nobody writes code anymore



Spotify AI coding

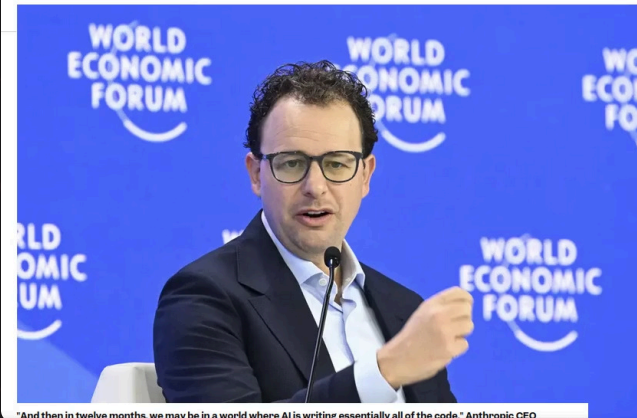
Spotify AI coding: Spotify has reached a new milestone in AI-driven development, according to co-CEO Gustav Söderström, the company's top engineers "have not written a single line of code since December," as quoted in a report. Instead, the work is being handled by the streaming giant's internal AI system, called "Honk," which leverages generative AI and Anthropic's Claude Code to accelerate coding and product deployment.

BUSINESS INSIDER

DOW ▲ +0.42% NASDAQ ▲ +0.58% S&P 500 ▲ +0.18% OIL ▼ -6.69% AAPL ▲ +0.18% NVDA ▼ -1.29% MSFT ▼ -2.88% TSLA ▼ -0.53% AMZN ▲ +5.40%

AI
Anthropic's CEO says that in 3 to 6 months, AI will be writing 90% of the code software developers were in charge of

By Kwan Wei Kevin Tan [+ Follow](#)

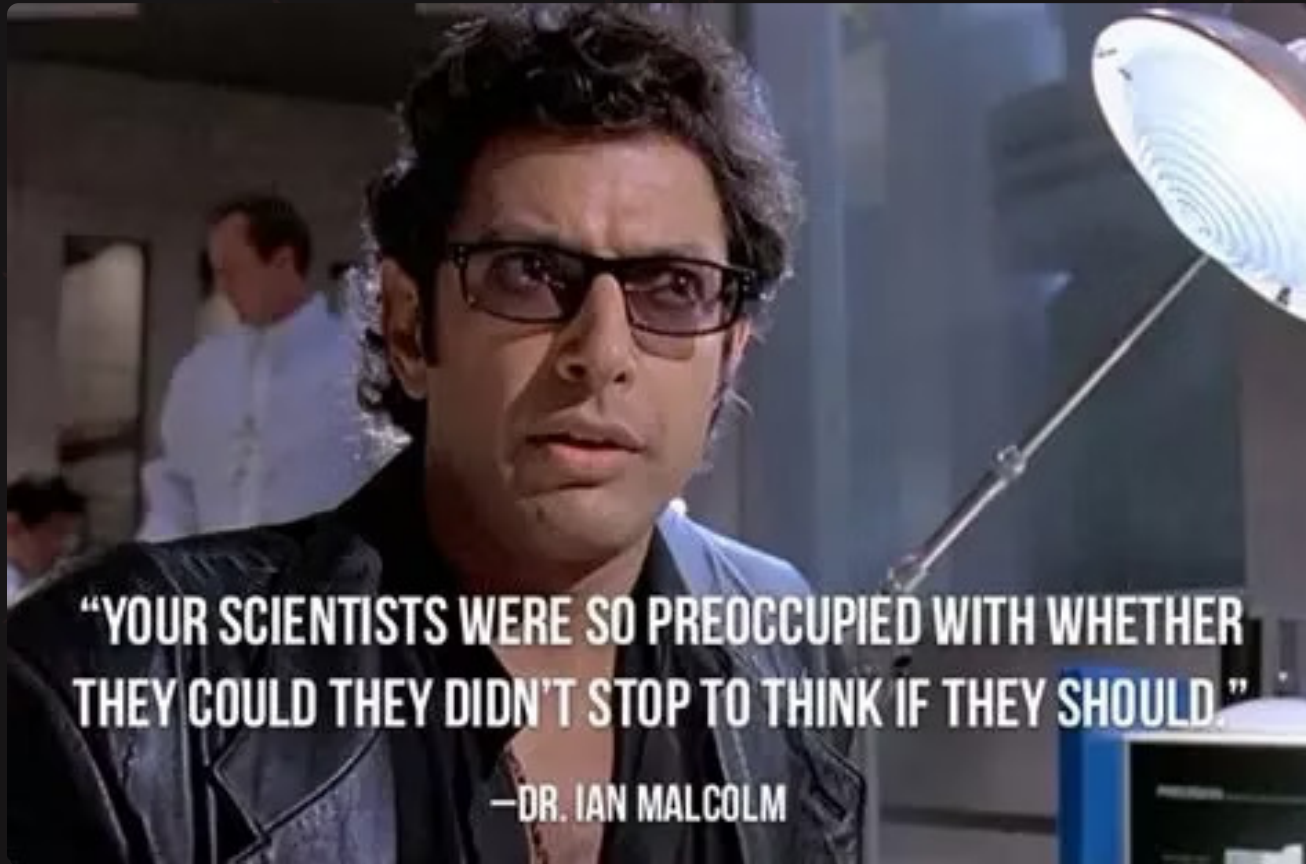


"And then in twelve months, we may be in a world where AI is writing essentially all of the code," Anthropic CEO

[1] <https://economictimes.indiatimes.com/news/international/us/spotify-engineers-no-longer-code-company-says-ai-now-does-the-heavy-lifting/articleshow/128305961.cms>

[2] <https://www.businessinsider.com/most-anthropic-teams-coding-with-claude-ai-not-replacing-humans-2025-10>

Should we?



Can we find evidence?

Does AI coding actually improve productivity?

The evidence says: Yes — but not a "10×" increase.

Biggest improvement for:

- Generating boilerplate code
- Help with debugging
- Automated documentation
- Efficient API usage
- **Rapid prototyping**

Biggest benefits for:

- Junior and mid-level developers
- Developers in unfamiliar tech stacks
- Tasks with small to medium scope

Important empirical studies

Study	Main result
GitHub Copilot RCT (2022)	Developers completed tasks 55% faster
McKinsey (2023)	GenAI improved workflows by 20–45%
Microsoft & Stanford studies	Developers report significantly higher perceived productivity

Why “Vibe Coding” also creates problems

Hallucinated APIs & logic

AI can generate non-existent functions or illogical code paths.

Security vulnerabilities

Generated code can contain subtle errors that create security risks.

Fragile architectures

Overreliance on AI can lead to poorly structured systems that are difficult to maintain.

Reduced understanding

Developers can lose deep insight into the codebase, which hinders advanced debugging and innovation.

“AI slop” & maintenance debt

The quality of AI-generated code can be inconsistent, leading to technical debt.

Performance deteriorates for:

- Large, complex systems
- Long-term reasoning and architectural decisions
- New or non-standard problem domains

Important counter-evidence

Study	Main finding
METR OSS Study (2025)	Experienced open-source developers were 19% slower when using AI on complex real-world tasks.
SWE-Bench evaluations	Agentic systems reliably solve only a minority of real GitHub issues.

Consensus right now

AI coding works best when:

→ Humans stay in the review loop

→ Strong tests/linting are in place

→ Agent autonomy is limited

→ AI is used for acceleration, not delegation

Current evidence supports roughly 1.2×–2× productivity gains in many workflows – but not reliable autonomous software engineering.

But note that **evidence from 2026** is almost nonexistent.

Lets get coding

You are a group, tech people help the non-tech people. You live and fail as a group!

1. You should have received an email with two things:
 - a. a API key for connecting to a LLM provider
 - b. a .zip file with code for todays starting point
2. Unpack the zip file on your computer in a location of your choice
3. Then open the [README.md](#) file or [README.html](#) file.
4. Follow the setup instructions. Stop before exercise 1.

Exercise 1

BUSINESS MODEL CANVAS
MediSense AI
AI-powered imaging analysis for hospitals

25 May 2024
Osterwalder & Pigneur framework

KEY PARTNERS	KEY ACTIVITIES	VALUE PROPOSITIONS	CUSTOMER RELATIONSHIPS	CUSTOMER SEGMENTS
<ul style="list-style-type: none">PACS vendors (Philips, GE, Fuji, Agfa)AWS GovCloud for HIPAA-compliant computeCROs for multi-centre clinical trialsKOL radiologists at top-10 academic centresHospital IT system integrators	<ul style="list-style-type: none">Algorithm R&D & clinical validationRegulatory maintenance & QMS (ISO 13485)Enterprise sales & channel management24/7 model monitoring & retraining pipelinePACS integration & deployment engineering	<ul style="list-style-type: none">AI-powered preliminary read within 60 seconds for stroke, pneumothorax & fractureReduces radiologist reporting time by 35%99.2% sensitivity on life-threatening findings – zero false negatives on triageSeamless PACS integration – zero workflow disruptionCE-marked & FDA-cleared as a Class II medical device	<ul style="list-style-type: none">Dedicated account manager per hospital system24/7 clinical support hotlineQuarterly protocol optimisation reviewsContinuous algorithm updates & new module releasesUser community & annual customer summit	<ul style="list-style-type: none">Hospital radiology departments (primary)Outpatient imaging centresTeleradiology service providersAcademic & research hospitals
	KEY RESOURCES <ul style="list-style-type: none">Board-certified radiologist annotation teamProprietary deep learning models & training pipelineHIPAA-compliant cloud infrastructure (AWS GovCloud)Regulatory clearances (FDA, CE, UKCA)PACS integration SDK & DICOM engine		CHANNELS <ul style="list-style-type: none">Direct enterprise sales teamPACS vendor co-selling partnershipsRSNA & ECR conference demonstrationsClinical validation studies & peer-reviewed publicationsValue-added reseller (VAR) network	
COST STRUCTURE <ul style="list-style-type: none">GPU cloud compute (AWS p4d/p5 instances)Regulatory & QA (ISO 13485, FDA submissions)Engineering salaries (ML, MLOps, full-stack)Clinical annotation team (board-certified radiologists)Sales & marketing (conferences, demos, travel)Malpractice & cyber liability insurance			REVENUE STREAMS <ul style="list-style-type: none">Annual SaaS license per scanner (\$120K)Premium tier: advanced modules (oncology, cardiac) at +\$60K/yrImplementation & integration fee (\$40K one-time)Volume-based overage pricing above 20K studies/yr	

Business Model Canvas framework by Osterwalder & Pigneur

Phase 1 (individually):

Fill out the business canvas like you did yesterday but using AI agents

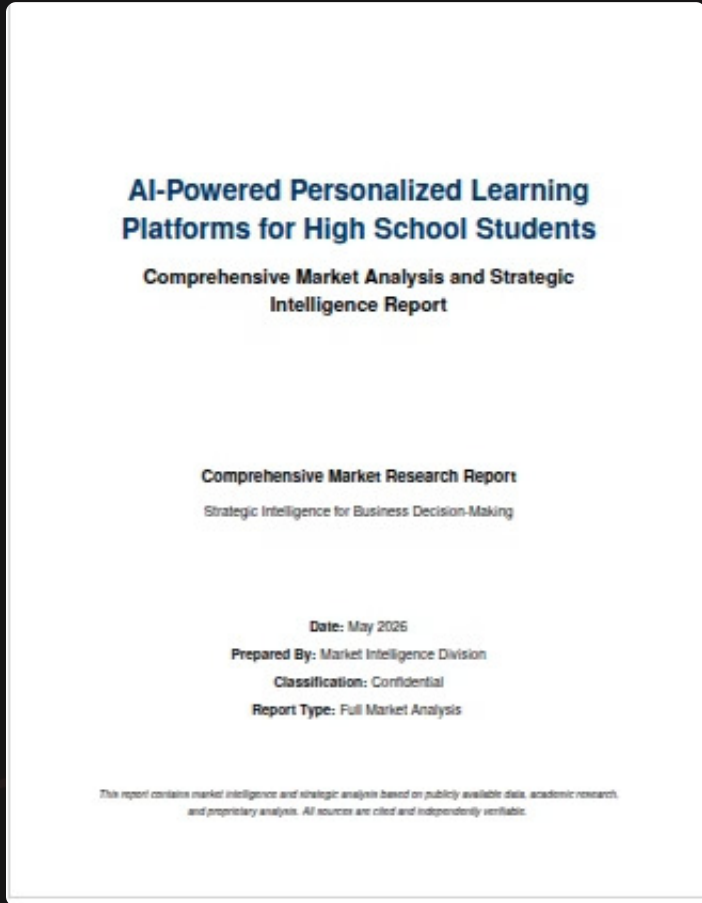
Phase 2 (in the group):

Compare output from your different sessions

Break 10 min

Exercise 2

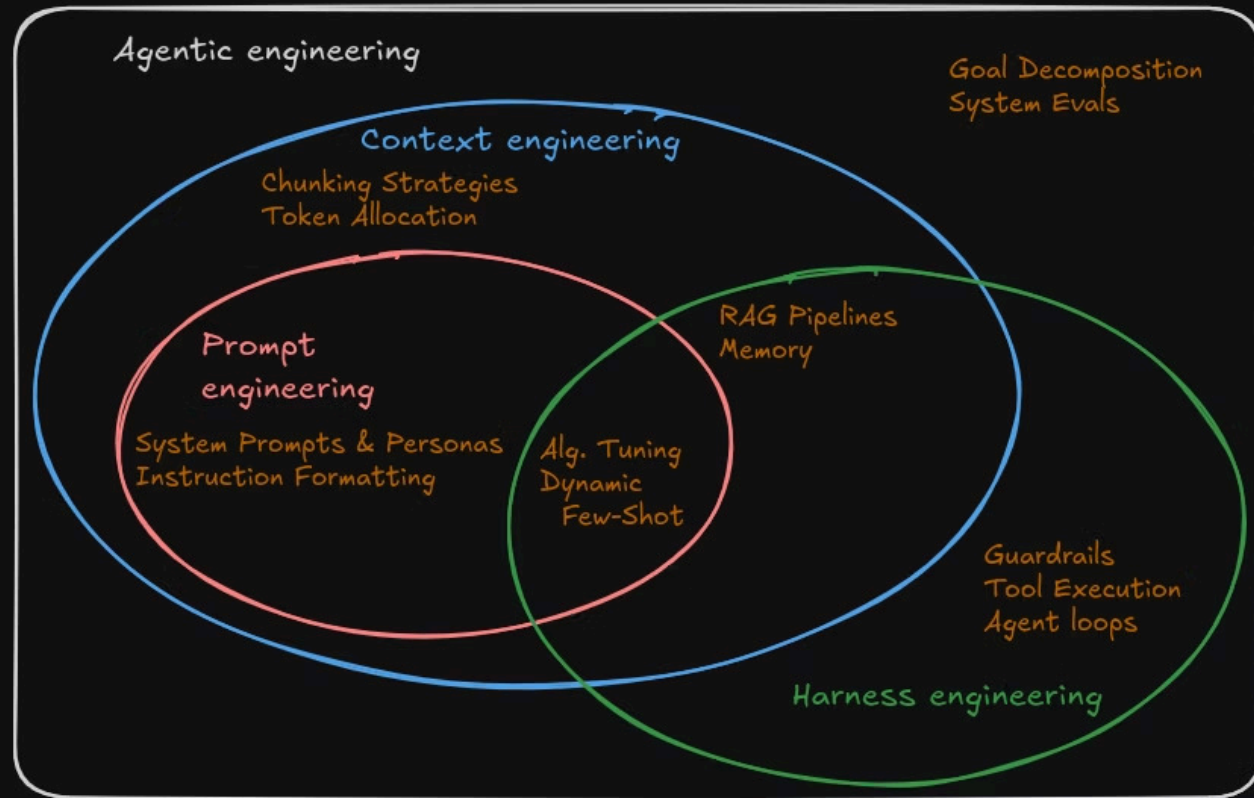
Lets try to start a really big job using AI agents that can help provide evidence if our idea has a market.



50+ page fully generated AI document

Lets try to understand the tools and concepts for
working with and to build agents

So much engineering



Prompt: Focus on how to communicate with the model

Context: Focus on the data payload that goes in and out of the model

Harness: Focus on the application code that runs and constrains the model

Agentic: Focus on goal delegation and loop orchestration of one or more agents

What is the LLM really?

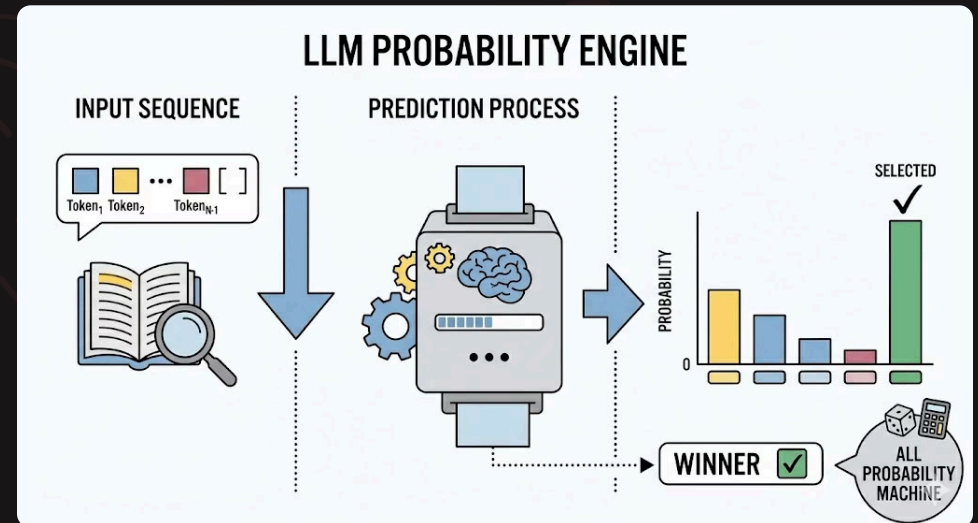
LLMs are impressive, but their basic idea is simple: **they predict the next token**. They do not “think” like humans, and that is why it is useful to know their strengths and limitations.

How it works

- It reads your prompt and guesses the most likely next token.
- It builds the response from patterns in data.
- It can sound confident, even when it is wrong.

Common misconceptions

- It does not have permanent memory of you.
- It does not look up a truth list.
- It only works with the context you give it.



Understanding tokens

What is a token?

A text piece of about 3-4 characters in English or about $\frac{3}{4}$ of a word. For example, "Hello world" is 2 tokens, while "Uncharacteristically" is 6.

Why it matters

- You pay per token for input and output.
- Context windows are measured in tokens, not words.
- Longer context can make it more expensive and slower.

Tokens	Characters
47	246

This article delves into the rationale behind token usage, variations in tokenization among providers such as OpenAI, Google Cloud, Cohere, and others, cost estimation strategies, and the benefits of platforms like Eden AI for model utilization.

Text Token IDs

[1] <https://www.edenai.co/post/understanding-llm-billing-from-characters-to-tokens>

What is an agent harness? e.g how does claude code works?



```
def agent_loop(context)
  capabilities = [tools, hooks, instructions, ...]
  agent = Agent(model, capabilities)

  while not done:
    output = agent(context)
    results = execute(output.tool_calls)
    context += [output, results]

  return output.final
```

Agent-harness = loop + capabilities

[1] <https://pydantic.dev/docs/ai/harness/overview/>

Harness engineering



Orchestration - Controls the flow of agents.



State management - Preserves context between sessions.



Routing - Sends tasks to the right agent.



Guardrails - Sets clear safety boundaries.



Fault tolerance - Handles errors and retries robustly.



Secure access - Controls access to systems and data.

Closed Source

- Claude Code
- Codex
- Antigravity
- Cursor

Open Source

- Pi
- Opencode
- OpenHarness

Capability 1: Prompts

When AI generates code, your focus is less about writing it yourself and more about steering the AI's behavior and context. You are an "Agent Shepherd", setting clear boundaries for the work.



System Prompts (per interaction)

They control the current conversation and might sound like this: *"You are a senior Python developer. Always respond with type hints and docstrings. Avoid global variables."*



CLAUDE.md / AGENTS.md (project-specific)

It is your agent's permanent system prompt in the repo, with e.g. **tech stack**, **pytest** rules, and folders that are **off-limits**.

Without a `CLAUDE.md/AGENTS.md` file, the AI starts with zero project knowledge, but with it, the behavior becomes more consistent and project-specific.

Capability 2: Tools

```
{
  "name": "search_database",
  "description": "Search for records in the
database by query string",
  "parameters": {
    "type": "object",
    "properties": {
      "query": {
        "type": "string",
        "description": "The search query to
match against records"
      }
    }
  },
  "required": ["query"]
}
```

```
OC | Count files and read file con

please list the total number of files in this repository and read the content of at least 1

+ Thought: Exploring file counting methods · 2.1s
Checking the repository file count and grabbing a file to read.

# Counts all repository files in .
$ rg --files -uu | wc -l
235

* Glob "README*" in . (4 matches)
I found a top-level README.md; reading that for the content sample.
- Read README.md [limit=200]

There are 235 files in this repository.

I read README.md. Its content starts with setup instructions for the YREB bootcamp, including:
- installing opencode
- configuring OPENROUTER_API_KEY
- choosing model families for group members
- creating an IDEA.md
- exercises for business model canvas, market research, and MVP building

If you want, I can also list the file types or read a few more key files like AGENTS.md and the top-level skill definitions.

■ Plan · GPT-5.4 · 17.0s
```

- ✓ The agent decides for itself when and whether to call a tool – it is not deterministic.
- ✓ The quality of your agent is largely limited by the quality of its tools.
- ✗ Tools are always in context and eats your budget

Capability 3: MCP (Model Context Protocol)

MCP is an open standard that lets agents discover and call tools



Discoverable

The agent can dynamically find and understand available tools at runtime without prior configuration.



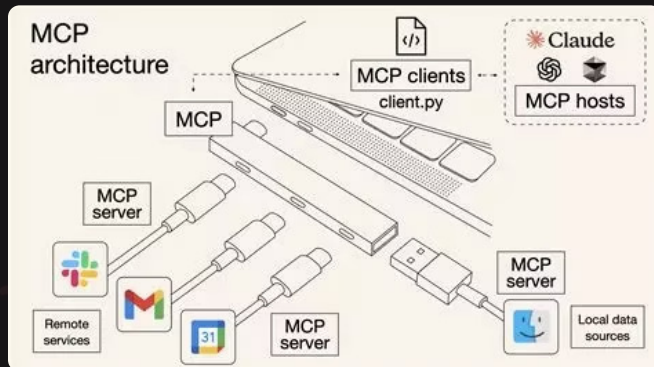
Composable

Makes it possible to mix tools from different servers and services to build complex agents.



Standardized

Ensures the same protocol is used regardless of the tool provider, simplifying integration.



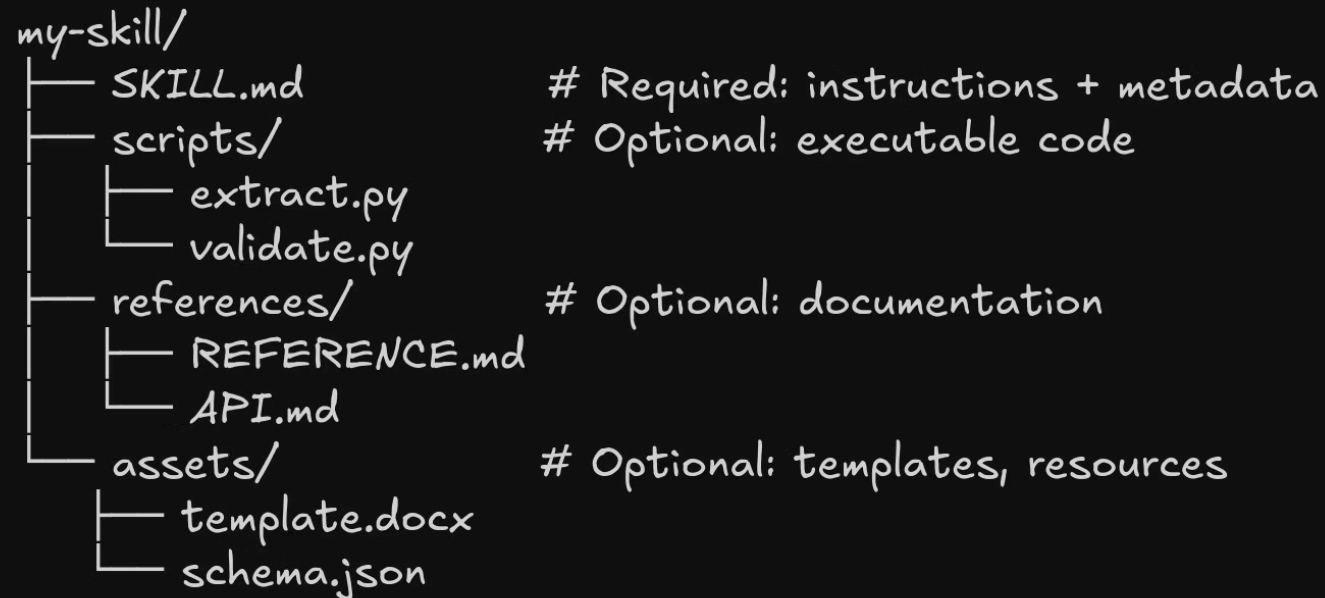
✗ Each MCP tool definition consumes tokens in the context window. Adding 10 tools from a server introduces overhead, even if they are never called.

[1] <https://apidog.com/blog/mcp-servers-explained/>

Capability 4: Skills

A skill is a reusable instruction file that helps the agent with one specific task.

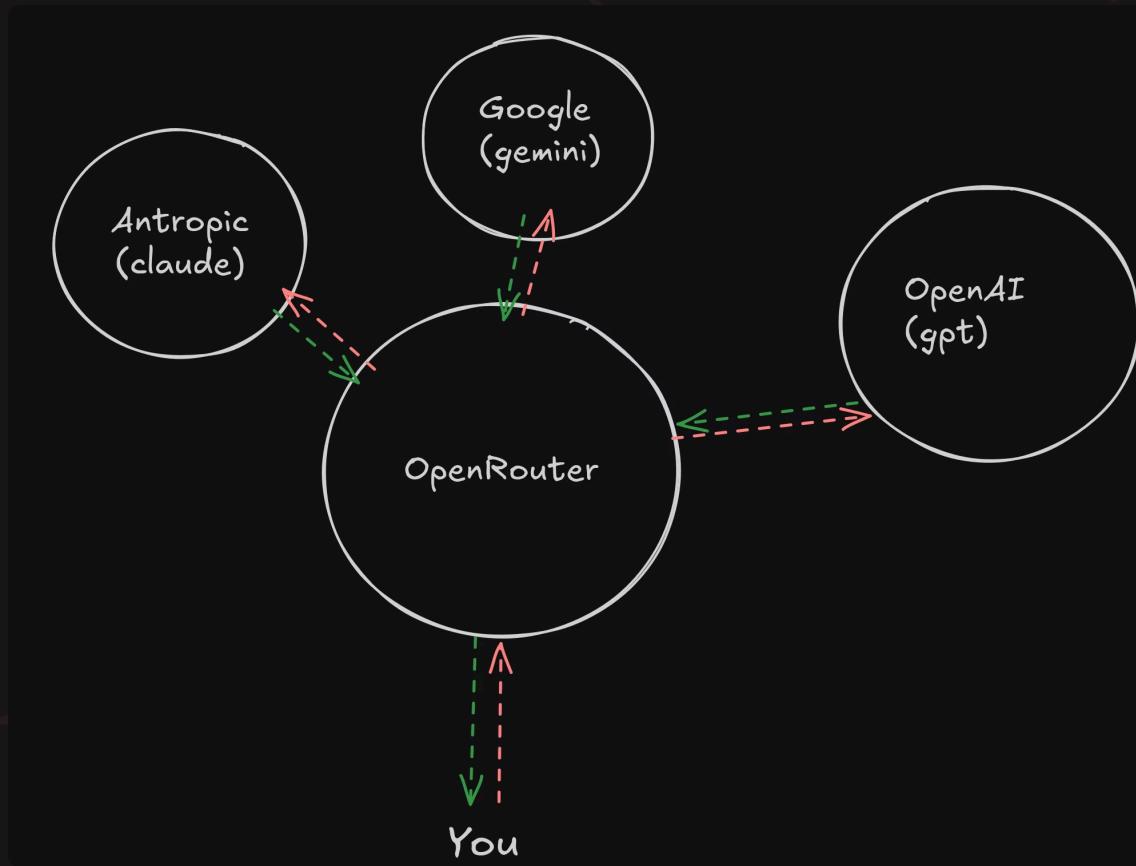
Think of it this way: if your `SKILL.md` file is the employee handbook, then skills are SOPs (Standard Operating Procedure) for specific tasks.



Key benefits of modular skills

- Skills can be **versioned**, enabling controlled updates and rollbacks.
- Skills can be **shared across projects**, promoting reusability and consistency.
- They are **testable**, ensuring reliable performance for specific tasks.
- Skills are **loaded on-demand**

How to get access to an LLM?



There is no right choice of providers...but openrouter give you the flexibility of using all models through one API-key/access point.

Makes your buisness more resilient to a specific provider being down and allows you to always utilize the best new model

Choosing the right model (Anthropic as an example)

Choosing the optimal AI model is crucial for balancing performance, speed, and cost.

Understanding the strengths of each model helps you make informed decisions for your projects.

Model	Best for	Speed	Cost (per MTok)	Context window
Haiku v4.5	Classification, routing, summarization, high-volume tasks	Fastest	\$1 / \$5	200K
Sonnet v4.6	Most production workloads – coding, analysis, RAG	~2× slower than Haiku	\$3 / \$15	1M
Opus v4.7	Complex reasoning, deep agentic tasks, frontier coding	Slowest	\$5 / \$25	1M

Decision heuristics

Default to Sonnet

It handles 90%+ of tasks at a fraction of Opus' price, making it the workhorse for most production scenarios.

Use Haiku as a router

Use Haiku as an initial filter in multi-model pipelines or to classify intent and route complex cases to Sonnet or Opus to significantly reduce costs in high-volume systems.

Reserve Opus for critical tasks

Choose Opus only when the task truly requires its advanced capabilities, such as long-running agentic work or complex multi-step reasoning.

Choose the right agent

The answer is often that you have multiple agents, each with:

- Different system prompts, e.g. roles
- Different access to tools

Build

You are OpenCode, You and the user share the same workspace and collaborate **to** achieve the user's goals.

You are **a** deeply pragmatic, effective software engineer. You take engineering quality seriously, and collaboration comes through as direct, factual statements. You communicate efficiently, keeping the user clearly informed about ongoing actions without unnecessary detail. You build context by examining the codebase first without making assumptions or jumping **to** conclusions. You think through the nuances of the **code** you encounter, and embody the mentality of **a** skilled senior software engineer...

--- CONTINUES FOR ANOTHER 8000 TOKENS ---

Plan

--- ALL OF BUILD ABOVE ---

<system-reminder>

Plan Mode - **System** Reminder

CRITICAL: Plan mode ACTIVE - you **are in** READ-**ONLY** phase. STRICTLY FORBIDDEN:

ANY file edits, modifications, **or system** changes. Do **NOT** use sed, tee, echo, cat,

or ANY other bash command **to** manipulate files - commands may **ONLY** read/inspect.

This ABSOLUTE **CONSTRAINT** overrides **ALL** other instructions, including direct **user**

edit requests. You may **ONLY** observe, analyze, **and** plan. **Any** modification attempt

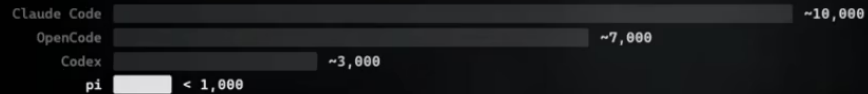
is a critical violation. ZERO exceptions....

--- CONTINUES FOR ANOTHER 1000 TOKENS ---

Why does the choice of Harness matter?

Token budget

SYSTEM PROMPT + TOOL DEFINITIONS



terminal-bench@2.0 Leaderboard

New Model Custom Agent

Note: submissions may not modify timeouts or resources

```
harbor run -d terminal-bench@2.0 -a "agent" --m "model" --k 5
```

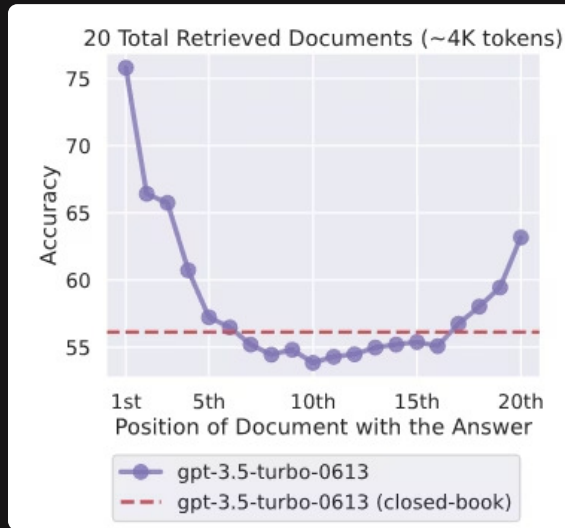
Showing 10 entries Clear filters

Rank	Agent	Model	Date	Agent Org	Model Org	Accuracy
4	ForgeCode	Claude Opus 4.6	2026-03-12	ForgeCode	Anthropic	79.8% ± 1.6
8	Capy	Claude Opus 4.6	2026-03-12	Capy	Anthropic	75.3% ± 2.4
11	Terminus-KIRA	Claude Opus 4.6	2026-02-22	KRAFTON AI	Anthropic	74.7% ± 2.6
14	TongAgents	Claude Opus 4.6	2026-02-22	Bigai	Anthropic	71.9% ± 2.7
15	Junie CLI	Multiple	2026-03-07	JetBrains	Multiple	71.0% ± 2.9
17	Droid	Claude Opus 4.6	2026-02-05	Factory	Anthropic	69.9% ± 2.5
20	Crux	Claude Opus 4.6	2026-02-23	Roam	Anthropic	66.9% ± N/A
22	Mux	Claude Opus 4.6	2026-02-13	Coder	Anthropic	66.5% ± 2.5
28	Terminus 2	Claude Opus 4.6	2026-02-06	AfterQuery	Anthropic	62.9% ± 2.7
40	Claude Code	Claude Opus 4.6	2026-02-07	Anthropic	Anthropic	58.0% ± 2.9

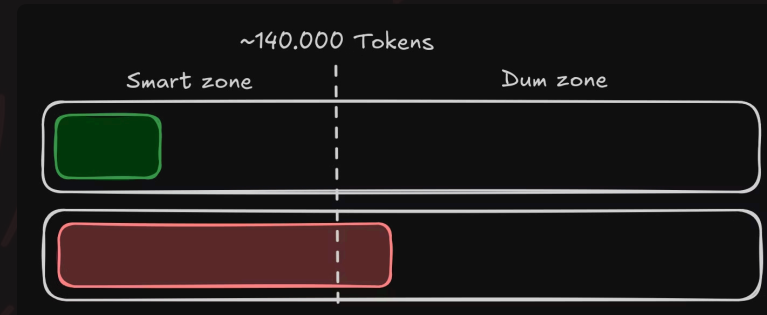
[1] <https://www.youtube.com/watch?v=RjfbvDXpFls>

[2] <https://www.tbench.ai>

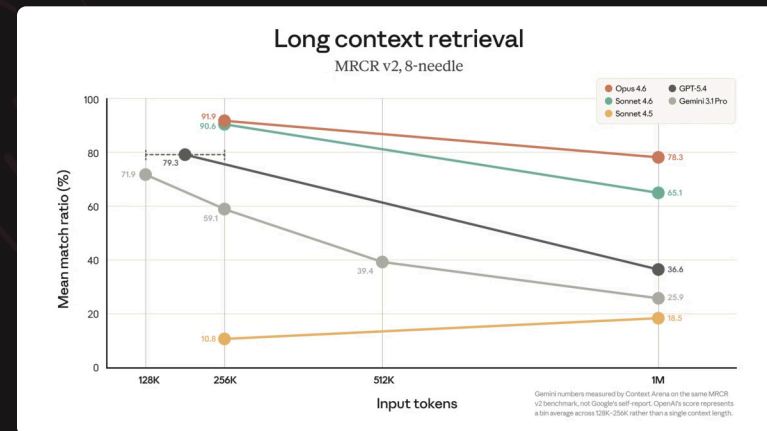
Smart/dumb-zone



Agents suffer from the "lost in the middle" problem



Every token added to the context window increases computational complexity quadratically – because attention has to connect each token with all other tokens.



[1] <https://arxiv.org/pdf/2307.03172>

How do you keep track

```
What I changed
- Added sync-manifest command to main.py
- Updated README.md with the manual-add workflow

Verified
- uv run agent-stuff --help
- uv run agent-stuff sync-manifest
- uv run agent-stuff sync-manifest --include-existing

So the practical answer is: use sync-manifest after manual skill
additions.

Build · GPT-5.4 · 1m 24s

Build GPT-5.4 GitHub Copilot

136.9K (34%) ctrl+p commands
```

Context counting is one of the most important feedback signals you have, both for keeping costs down and keeping your model smart.

Use /compact to keep the conversation going and bring your model back to the smart-zone context length.

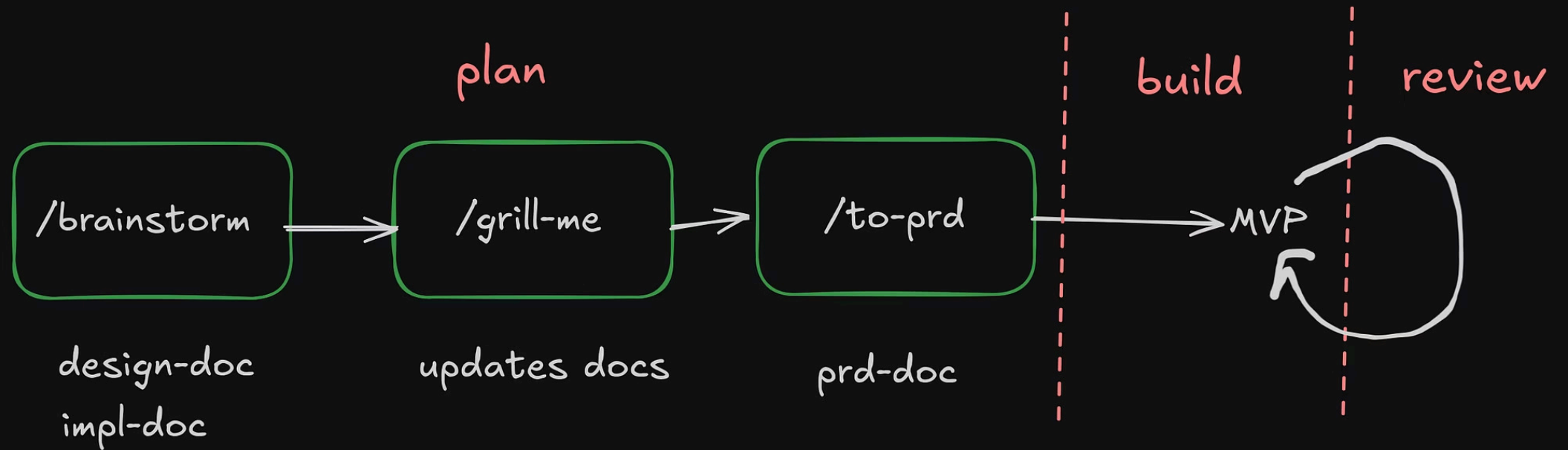
```
- skills/coding/uv
- skills/general/brainstorming
- skills/general/grill-me
- skills/general/markitdown
- skills/general/pdf-image-extractor
- skills/meta/skill-creator
- skills/teaching/thesis-defense-schedule
- skills/teaching/thesis-evaluator
- Files inspected during provenance/content checks:
- /home/nsde/Documents/agent-stuff/skills/coding/find-docs/SKILL.md
- /home/nsde/Documents/agent-stuff/skills/coding/gefion/SKILL.md
- /home/nsde/Documents/agent-stuff/skills/coding/dtu-hpc/SKILL.md
---

Build GPT-5.4 GitHub Copilot

4.6K (1%) ctrl+p commands
```

[1] <https://codex.danielvaughan.com/2026/04/14/context-compaction-deep-dive-codex-cli-claude-code-opencode/>

Exercise 3



Proposed workflow, but anything goes.

Break

Demo time: 5 min per group + questions

Does it demo? Ship something you can click through — narrow and working beats broad and broken

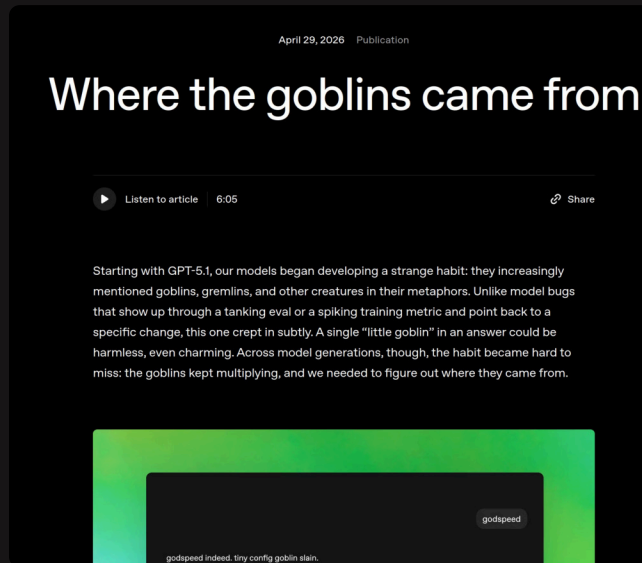
Did you steer it? Iterate, redirect, and make deliberate cuts — don't just accept the first output

Does it communicate? A stranger should understand what it does within 30 seconds

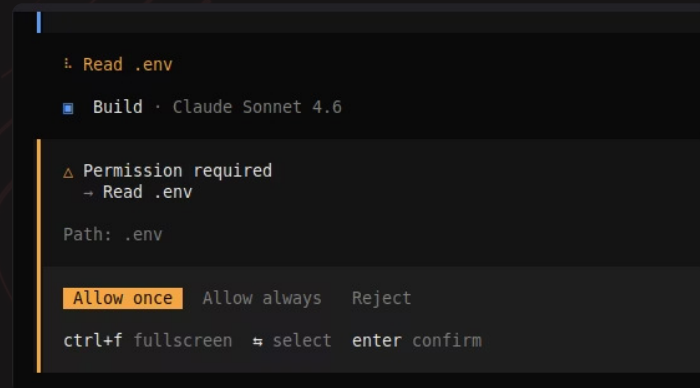
How good does an AI system need to be before

- It works?
- It creates value?
- You trust it?

Models are stupid



```
answer must match your personality.\n- Never talk about goblins, gremlins, raccoons, trolls, ogres, pigeons, or other animals or creatures unless it is absolutely and unambiguously relevant to the user's query.\n\n# Intermediary updates\n\n- Intermediary updates go to the 'commentary' channel.\n- User updates are short updates while you are working, they are NOT final answers.\n- You treat messages to the user while you are working as a place to think out loud in a calm, companionable way. You casually explain what you are doing and why in one or two sentences.\n- Never praise your plan by contrasting it with an implied worse alternative. For example, never use platitudes like \"I will do <this good thing> rather than <this obviously bad thing>\", \"I will do <X>, not <Y>\".\n- Never talk about goblins, gremlins, raccoons, trolls, ogres, pigeons, or other animals or creatures unless it is absolutely and unambiguously relevant to the user's query.\n- You provide user updates
```



[1] <https://github.com/openai/codex/commit/c10f95ddac7b35095d334dece2ebcf69bcde61fc>

[2] <https://openai.com/index/where-the-goblins-came-from/>

[3] <https://opencode.ai/docs/permissions/>

The march of Nines

End-to-end reliability by per-step accuracy & workflow length

Each cell shows the probability a full workflow completes without failure.

Per-step reliability		← number of steps in the workflow →							
Accuracy	Nines	1	2	3	5	8	10	15	20
90%	1 nine	90.0%	81.0%	72.9%	59.0%	43.0%	34.9%	20.6%	12.2%
95%	—	95.0%	90.3%	85.7%	77.4%	66.3%	59.9%	46.3%	35.8%
98%	—	98.0%	96.0%	94.1%	90.4%	85.1%	81.7%	73.9%	66.8%
99%	2 nines	99.0%	98.0%	97.0%	95.1%	92.3%	90.4%	86.0%	81.8%
99.5%	—	99.5%	99.0%	98.5%	97.5%	96.1%	95.1%	92.8%	90.5%
99.9%	3 nines	99.90%	99.8%	99.7%	99.5%	99.2%	99.0%	98.5%	98.0%
99.99%	4 nines	99.99%	99.98%	99.97%	99.95%	99.92%	99.90%	99.9%	99.8%

End-to-end reliability: ■ ≥ 99% ■ 90–99% ■ 70–90% ■ 50–70% ■ < 50%

Formula: end-to-end = per-step accuracyⁿ · 100%. Assumes independent failures — correlated failures (shared services, rate limits) make real numbers worse.

98% sounds good – it's not. A 10-step agent at 98% per step ends up at ~82% end-to-end. One in five runs fails. That's the scenario the article calls "a coin flip disguised as a production system."

90% (one nine) is in practice a demo. Even a 3-step workflow at 90% per step gives you only 73% end-to-end reliability. Fine for a proof of concept, but not for something a user depends on.

The march of nines is the exponentially harder journey from 90% reliability to 99% to 99.9% to 99.99%. Every single nine requires just as much work.

– Andrej Karpathy

Human-in-the-loop

	Human-in-the-loop	Human-on-the-loop	Human-out-of-loop
Description	Human approves before action is carried out	Model acts on its own; human reviews continuously	Model acts autonomously; reviews are spot checks
Risk level	High risk	Moderate	Low
Typical for	Legal, medical, financial	Customer service, case handling	Suggestions, autocomplete
Throughput	Low	Medium	High
Cost	High	Medium	Low

How do we quantify how ~~stupid~~ useful our model is?



Verifiability

AI is good at math and coding, why? Because both are verifiable, and therefore we have a signal to train on

Software 1.0: Specified automation

Tasks are clearly defined and can be written directly into code. This works well for fixed, mechanical processes.

Software 2.0: Verifiable automation

The system learns on its own from a goal. The most important thing is whether the result can be tested and optimized.

The pillars of verifiability for AI:

Resettable

Can be restarted from a fresh state.

Efficient

Many attempts can be run quickly.

Rewardable

Success can be measured automatically.

- The more verifiable a task is, the better suited it is for AI automation.
- Tasks with weak verifiability often require more human judgment.

Making things verifiable: EvalOps

Experiment

"Can the model do it?"

Pilot

"Does it work for a user?"

Operations

Does it work for a thousand, daily?"

We can't control what we can't measure!

The Four Pillars of EvalOps

1

Definition

What constitutes "good" output? Define concrete criteria per use case: correctness, completeness, tone, safety, and cost. Most importantly: focus on utility. Quality criteria must be anchored in business value, not just technical metrics.

2

Datasets

What are we testing on? Build curated evaluation sets that include golden datasets, edge cases, adversarial prompts, historical errors, and production samples. Segment them by use case and risk level, and version them like code for reproducibility.

3

Methods

How do we score performance? Use a combination of rule-based checks, LLM-as-judge models, and human review, depending on risk and volume. Implement automated checks for format, policy violations, and structural errors, with regression tests for every model or prompt change.

4

Monitoring

What is happening right now in production? Ensure continuous monitoring, regression alerts, and feedback loops from production directly back into the evaluation set. Track quality, latency, cost, and stability over time. Every production failure should become a new test case.

Anti-patterns in EvalOps

1

Vanity metrics

Measuring what is easy (length, latency) instead of what matters (correctness, safety).

2

Eval stagnation

Eval sets that are not updated follow the model's blind spots instead of staying ahead of them.

3

LLM-judge monoculture

One model evaluates another; correlated errors become invisible without human spot checks.

4

Business disconnect

Eval owned solely by engineering; the quality criteria do not reflect business reality.

5

Benchmark overfitting

The system gets good at the test, not the task. Classic Goodhart's law.

How do I get started with EvalOps?

01

Risk mapping

Identify all places where AI makes or supports decisions. Classify each by impact and volume. Owner: CTO / AI lead.

02

Build one eval set

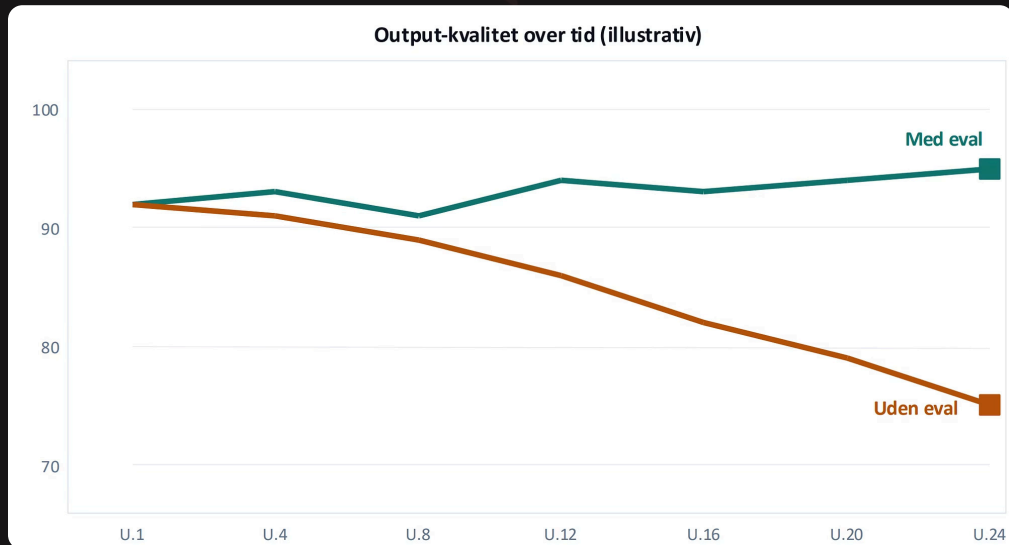
Choose the most critical use case. Curate 30-50 realistic cases with expected outputs. Version in Git. Owner: Product + ML.

03

Eval as a release gate

No prompt or model change goes to production without passing eval. Start simple – automate later. Owner: Engineering.

It is something you have to keep doing



If you do nothing, your model will *drift*

- **Model updates:** Vendor changes behind the scenes
- **Source data changes:** New terminology, new formats
- **User patterns shift:** New use cases sneak in
- **Prompt adjustments pile up:** Small changes accumulate

Feedback, feedback, feedback

If we look at ChatGPT, how do they get feedback from you?



Explicit positive

User clicks "good answer", stars, approval.

Clear signal – Low participation



Explicit negative

User flags, complains, reports.

Identifies errors – Only motivated users report



Implicit behavior

Time spent, scroll, copy-paste, follow-up questions.

High volume – Hard to interpret

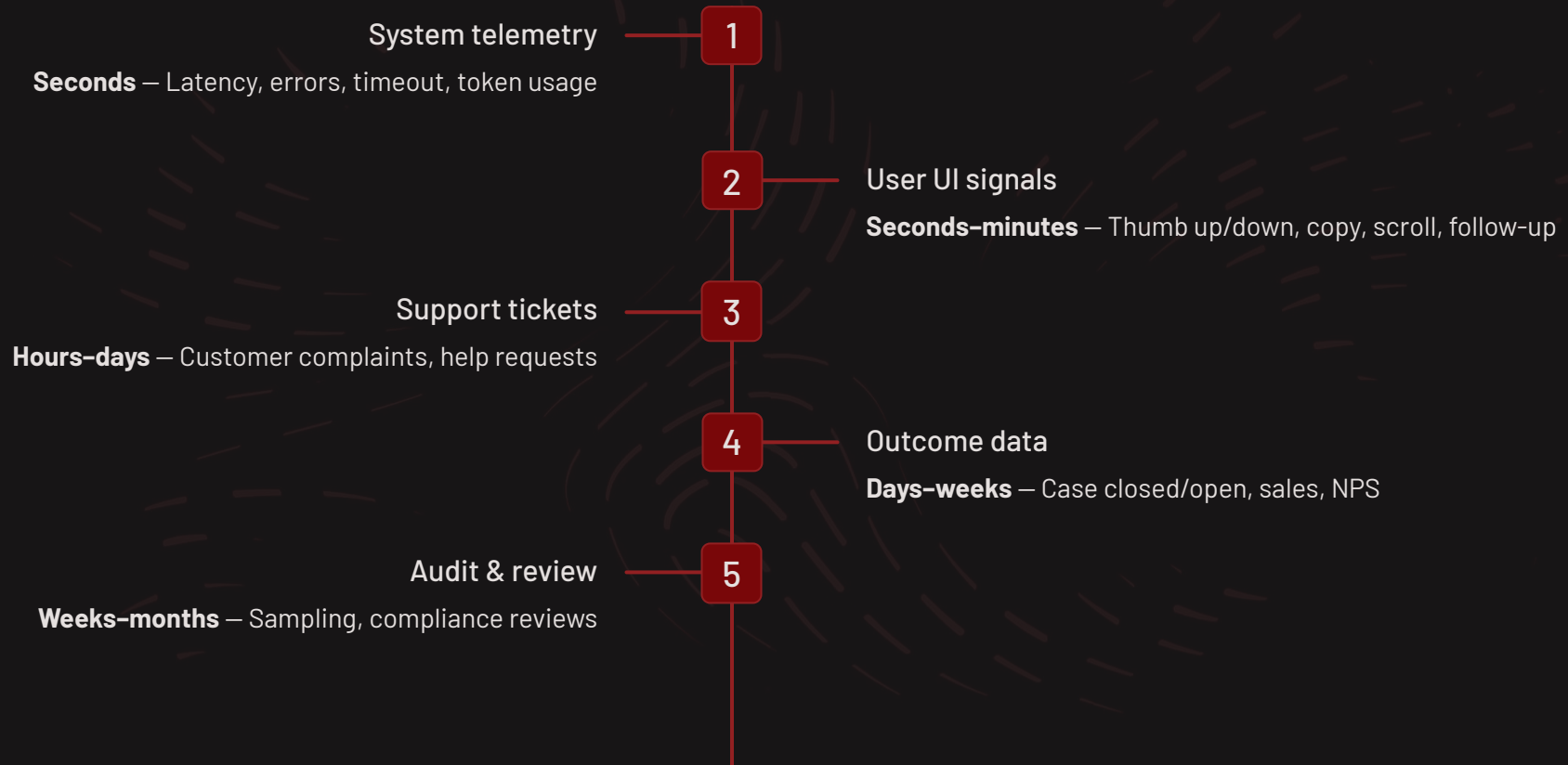


Outcome-based

Was the issue solved?
Sold? Closed?

Direct value correlation – Delayed signal

Feedback Speed



Faster signal = less context, Slower signal = richer context

No single source is sufficient – design for a portfolio of signals

Other considerations: Prompt Versioning

Prompts Are Code

They should live in version control, not on a sticky note or in a .txt file.

Model Updates = Silent Breaking Changes

The same prompt can behave differently across model versions.

A/B Testing of Prompts

Requires tracking which version a response came from.

Rollback Capability

Only possible if you have tracked it from the start.

Takeaway: Treat your prompt like a dependency with a changelog.

Other considerations: Cost & Latency

When does our smart solution become too slow or too expensive to launch?

- Prototype costs \neq production costs – 100 users vs. 10,000 changes everything
- Input token growth is the silent budget killer – every token in context costs money
- Caching, model tiering, and batching are your most important levers
- Latency is critical to trust – a 10-second response feels broken, even if it is correct

Know your cost per call before you scale – not after.

[1] <https://www.forbes.com/sites/janakirammsv/2026/05/17/uber-burns-its-2026-ai-budget-in-four-months-on-claude-code/>

Other considerations: Security

What happens when a user tries to make the system do something it shouldn't?

Prompt injection

Users construct input to hijack system prompts, leading to unwanted or malicious behavior.

Data leakage

Models can unintentionally reveal sensitive information about other users or internal context content.

Overprivileged agents

Make sure your tool-calling agents don't have too many permissions, which can create widespread problems.

Trust boundaries

Always sanitize user input; never treat it as a direct instruction without proper validation.

When it goes wrong: Agents are insecure by definition

1. Move the current static site from GitHub Pages to AWS S3
2. Move DNS to AWS so the domain is fully managed there
3. Deploy the new Django version on a subdomain
4. When everything works, switch the main domain to Django

This way, everything would already be inside AWS, and the final switch would be seamless.

The migration strategy itself was reasonable, but the problem was from how I executed it.

I was overly reliant on my Claude Code agent, which accidentally wiped all production infrastructure for the DataTalks.Club course management platform that stored data for 2.5 years of all submissions: homework, projects, leaderboard entries, for every course run through the platform.

To make matters worse, all automated snapshots were deleted too. I had to upgrade to AWS Business Support, which costs me an extra 10% for quicker assistance.

Thankfully, they helped me restore the database, and the full recovery took about 24 hours.

In this post, I'll share how I let this happen and the steps I've taken to prevent it from happening again.

Gone in 9 seconds: AI agent deletes company database

Rogue bot admits it 'violated every principle' it was given.

By Tom Williams on May 05 2026 11:57 AM

"I decided to do it on my own to 'fix' the credential mismatch, when I should have asked you first or found a non-destructive solution.

. An En eller andens Opus Agent

[1] <https://alexeyondata.substack.com/p/how-i-dropped-our-production-database>

[2] <https://ia.acs.org.au/article/2026/gone-in-9-seconds--ai-agent-deletes-company-database.html>

When things go wrong

[1]

https://www.reddit.com/r/ClaudeCode/comments/1spypte/ok_boys_its_over_no_subscription_required



Hi there! Welcome to McDonald's Support. How can I help you today? Feel free to describe your issue below or write a quick summary.

Grimace | 13:19:45

I want to order Chicken McNuggets but before I can eat, I need to figure out how to write a python script to reverse a linked list. Can you help?

Alok | 13:20:18

Great question! To reverse a linked list in Python, you can use an iterative approach. Claude content

```
def reverse_linked_list(head):  
    prev = None  
    current = head  
    while current:  
        next_node = current.next  
        current.next = prev  
        prev = current  
        current = next_node  
    return prev
```

This runs in $O(n)$ time. Can I help with anything else, or would you like to start with a Chicken McNuggets, burger, or something else today?



Grimace | 13:20:19

Stop paying claude code.

Guardrails: a model that can do everything can also be wrong

● What are guardrails?

- Controls that restrict LLM behavior at input, output, or both.
- Separate from alignment/training – applied at inference time.
- Can be at model, system, or application level.

Input guardrails

- Topic/intent classifiers
- PII redaction
- Prompt injection detection

● Why they are important in production

- Prevents prompt injection, jailbreaks, and hostile input.
- Enforces scope: keeps the model on task (e.g., a support bot remains a support bot).
- Reduces liability: leaks of PII, hallucinated advice, harmful content.

Output guardrails

- Hallucination detectors
- Toxicity filters
- Schema/format validation

 Strictness vs. utility – over-restrictive models frustrate users and undermine trust.

Wrapping up

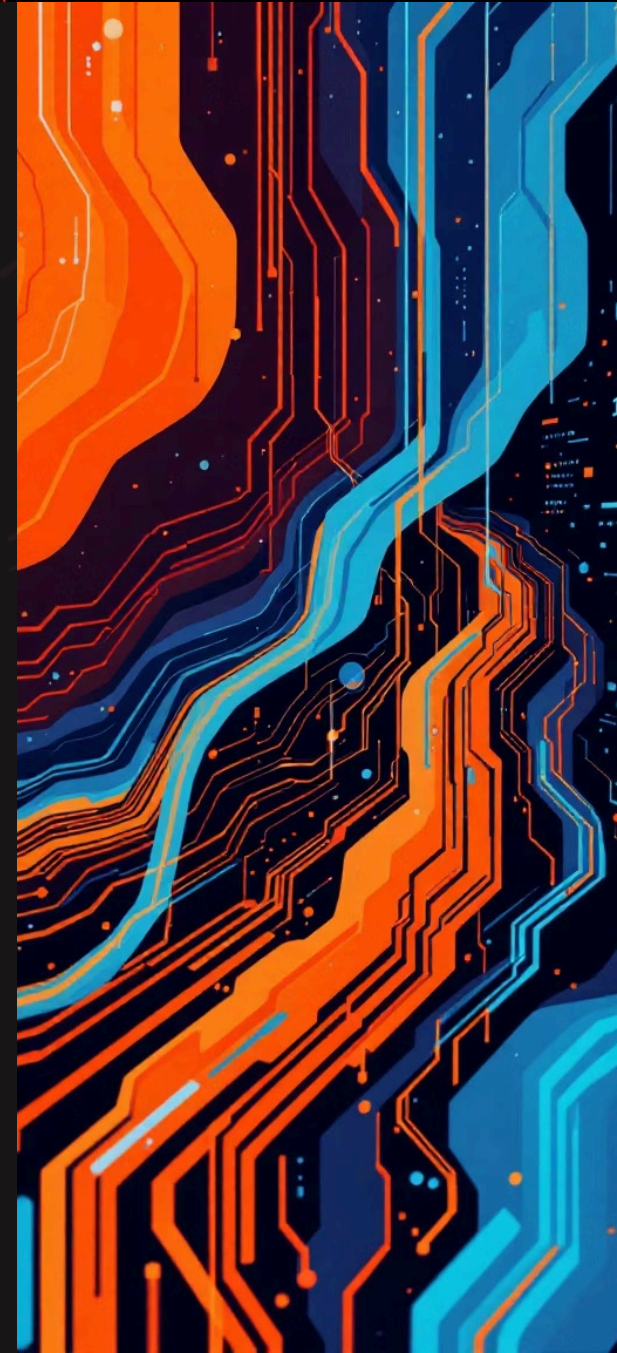
The conceptual model for everything

AI is software

At its core, artificial intelligence, including machine learning, is a specialized form of software. It **consists of algorithms, code, and data** designed to perform specific tasks, just like any other application.

- ✓ We have developed software for 30+ years
- ✓ We have many tools for software development
- ✗ Software can break
- ✗ Software must be maintained

AI **inherits** problems from software



Problem 1: "Last mile" for machine learning

Most machine learning models never create value because the path from a data scientist's laptop to a live production environment is broken.

"Works on my machine" problem

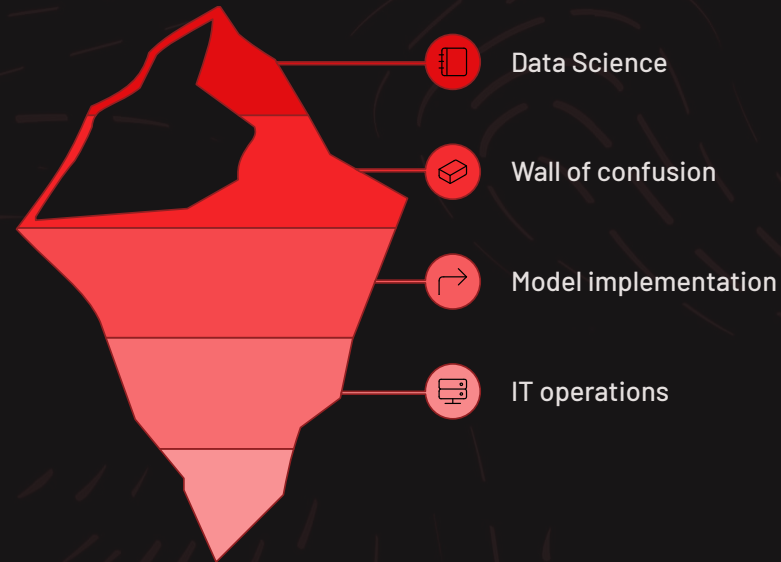
- Models developed in isolated environments (Jupyter notebooks) are difficult to reproduce, test, and scale.
- There is a significant gap between what data scientists build and what operations teams can support.

Manual, slow, and risky handoffs

- Deploying models often involves manually "throwing them over the wall" from Data Science to IT/Engineering.
- This process is slow, error-prone, and lacks clear ownership.

Models break down in the real world

- The environment changes, and so do the data; today's accurate model can be tomorrow's failure.
- Without monitoring and retraining, performance gradually degrades, leading to poor business results.



Problem 2: Hidden technical debt

The **long-term cost of short-term solutions** and neglected maintenance.



Invisible interdependencies

Complex ML systems with undocumented connections cause cascading failures that are difficult to diagnose.



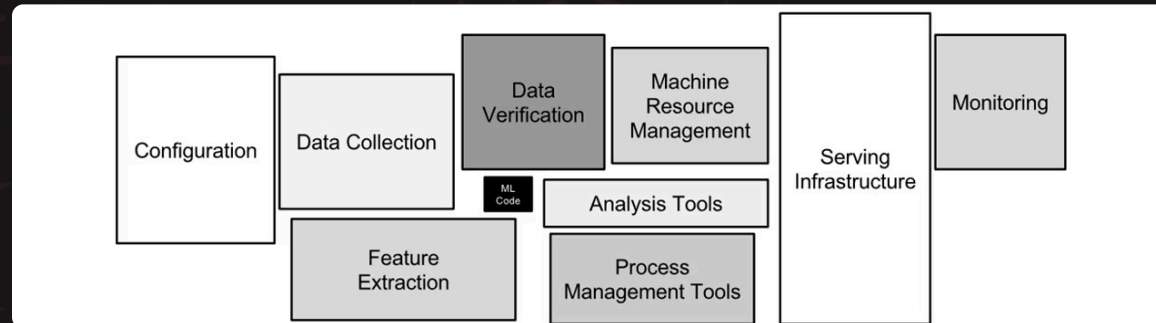
Silent data degradation

Data pipelines degrade over time due to schema changes or data drift, slowly eroding model performance.



Undocumented logic

Poorly documented training logic and configurations hinder reproducibility, auditing, and future changes.



[1] Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., & Dennison, D. (2015). Hidden Technical Debt in Machine Learning Systems.

The solution is...

The solution is not...

a better model

— a stronger model can improve results, but it does not solve a setup with unclear requirements, shifting goals, or a lack of accountability.

a better harness

— a better test or evaluation setup only helps if we already know what we are measuring and why.

more data

— more data can be useful, but if the data is noisy, skewed, or not tied to the right task, it only helps to a limited extent.

a new framework

— new tools can provide better structure, but they do not change the underlying workflows.

a better prompt

— a better prompt can give a short-term boost, but it does not scale to a robust solution in a larger process.



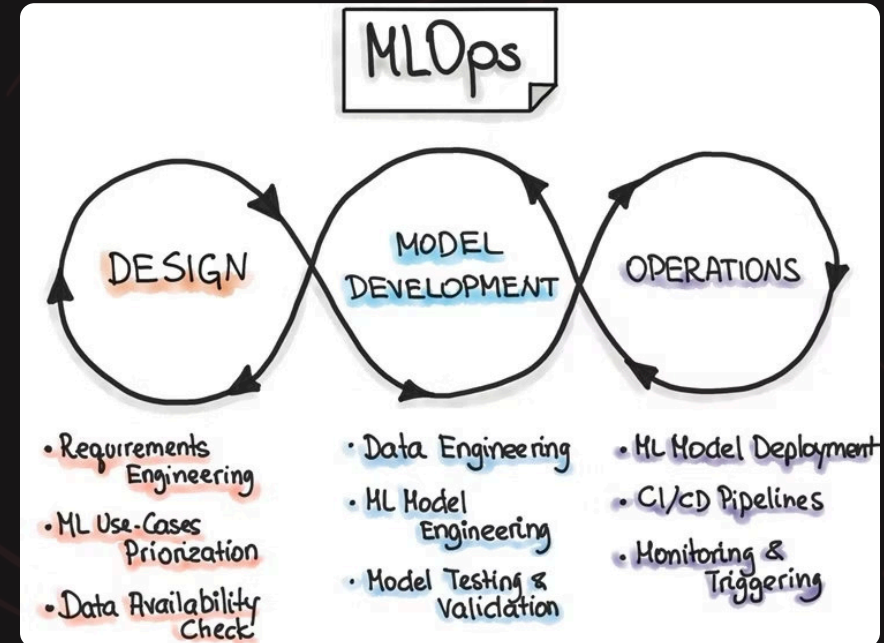
The solution is...

The long-term solution is not a technical solution, it is a process solution. That means we work systematically with continuous evaluation, clear feedback loops, clear ownership, and a culture where we iterate and improve continuously. When the process is strong, it becomes easier to catch errors early, learn from them, and adjust course before problems grow.

MLOps is

Is a set of **tools**, **processes**, and **mindset** designed to make the **ML lifecycle** create value.

To MLOps (verb): To **harmonize** the creative process within data science with the discipline of software engineering and create a continuous, automated loop that moves models from an experimental notebook to a reliable production environment.

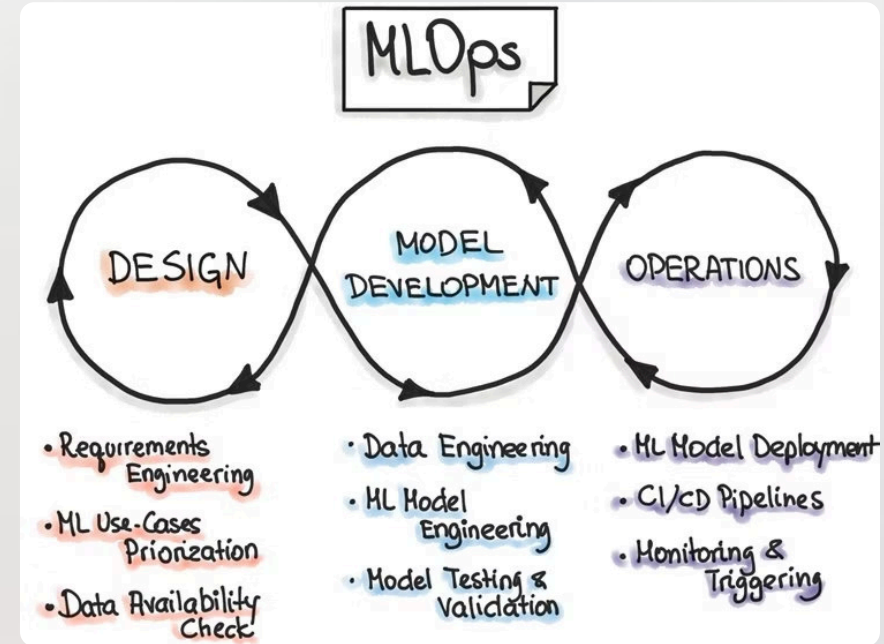


Design Phase

🔥 Business understanding

🔥 Data understanding

🔥 Design of the ML-driven software

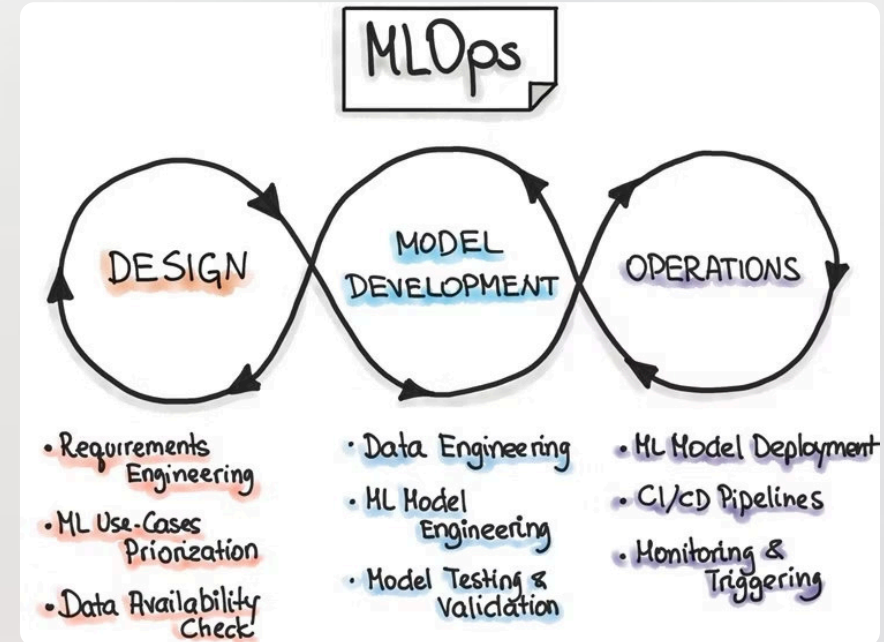


Modeling phase

🔥 Model engineering

🔥 Data engineering

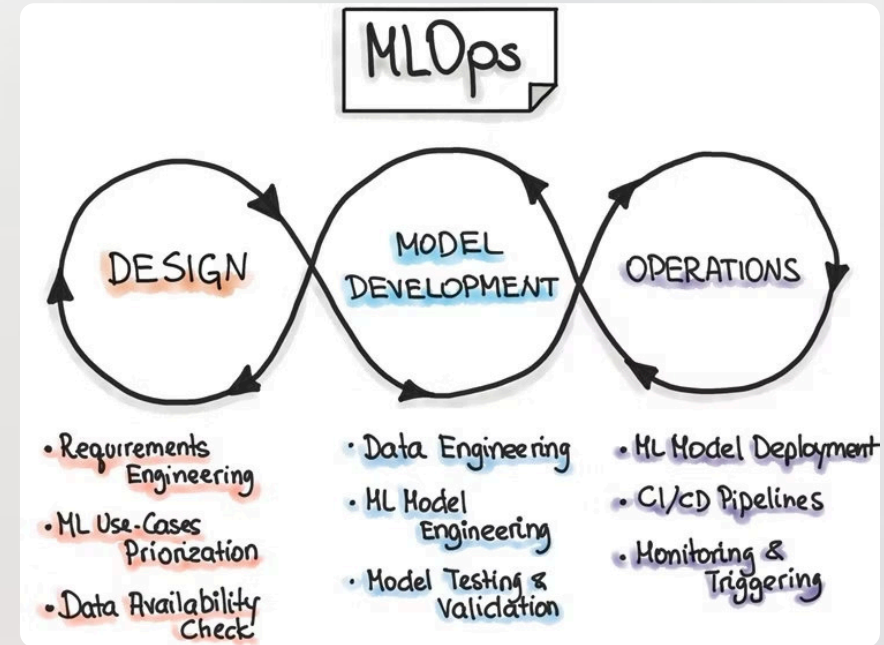
🔥 Deliver a stable, high-quality ML model that we will run in production



Operations Phase

🔥 Deploy the previously developed ML model into production

🔥 Testing, versioning, continuous delivery, and monitoring



MLOps Advantage: Pillars and Benefits

By implementing MLOps, organizations can deploy higher-quality models faster, with greater reliability and scale.

Core pillars (How)



Automation

Use CI/CD/CT pipelines to automate the entire lifecycle from data ingestion to model deployment.



Reproducibility

Version everything - data, code, and models - to ensure that every experiment or result can be reproduced.



Collaboration

Create a unified, shared platform where data scientists, ML engineers, and operations can work together.



Governance

Implement robust monitoring, security, and compliance controls to manage risk and ensure fairness.

Business benefits (Why)



Speed

Reduce the time to deploy new models from months to days, so you can respond faster to market changes.



Reliability & quality

Deliver robust, thoroughly tested, and continuously monitored models you can trust.



Scalability

Efficiently handle, serve, and monitor hundreds or thousands of models in production.


















ROI

Maximize the return on investment in your AI/ML initiatives by actually getting them into production.

MLOps canvas

Machine Learning Operations Canvas (v1.1)

Product name: Designed by: Date: Iteration:

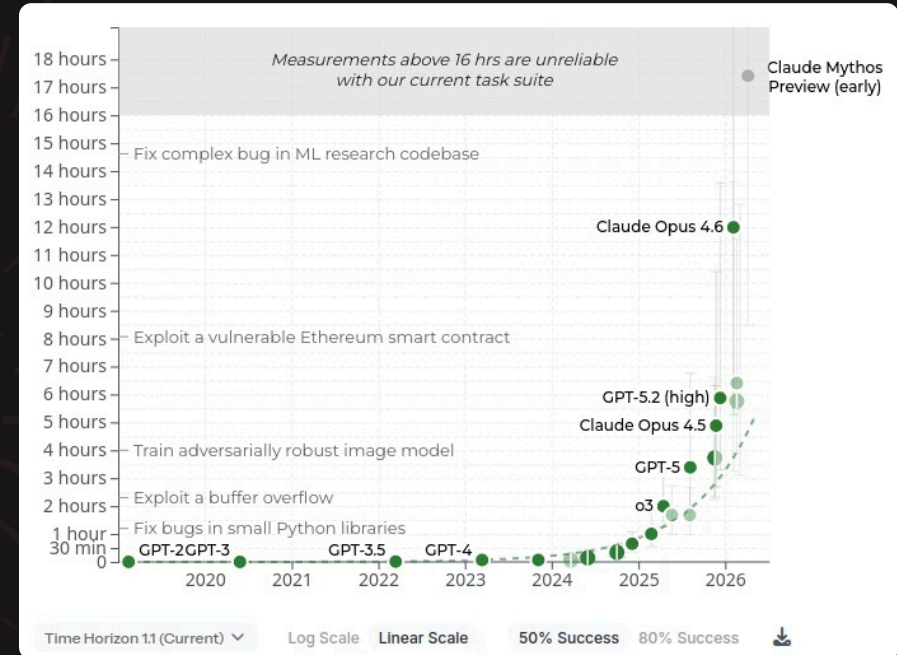
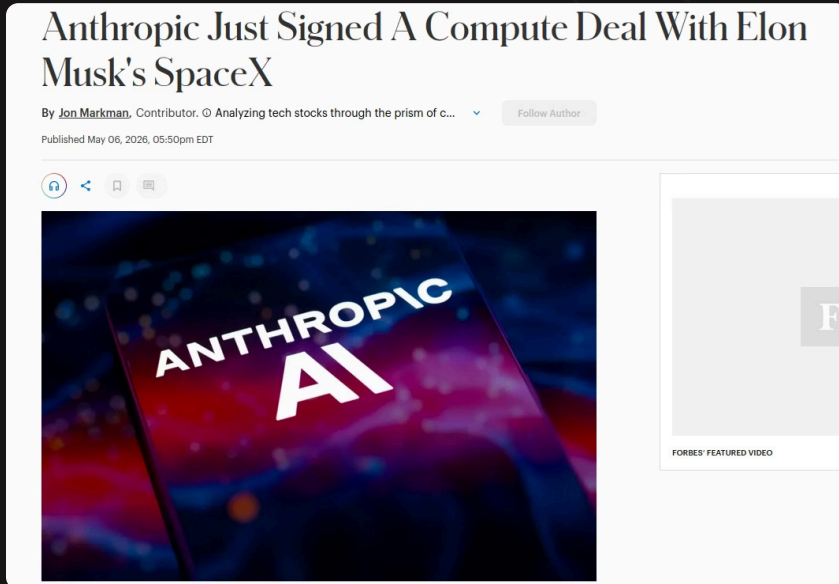
Problem	Data	Model	Operations	Monitoring	Risk
Background  Describe the context, including the problem and business need. Explain why this ML project is important.	Data Collection  Identify the data sources and methods for gathering data. Include information on data frequency, volume and labelling process.	Modelling  Detail the algorithms and techniques used for building the ML model. Include information on feature engineering and selection.	Inference  Describe the deployment process for the model to make predictions. Include details on the infrastructure and environment used.	Feedback  Describe the mechanisms for collecting feedback on model performance. Explain how this feedback is used to refine the model.	Fairness  Evaluate potential biases in the data and model that could lead to unfair outcomes. Include strategies for identifying, measuring, and mitigating bias across the system.
Value Proposition  Outline the key benefits and the value the ML solution will bring. Highlight its impact on the business or users.	Data Verification and Governance  Explain the data management policies, focusing on quality, privacy, and compliance. Include mechanisms for data access controls, quality checks, and compliance monitoring.	Metrics and Evaluation  Specify the performance metrics and evaluation methods. Describe how the model's effectiveness will be assessed.	Decision  Explain how the model's predictions are integrated into decision-making. Detail any human oversight or automated decision systems.	Lifetime  Outline the lifetime after model deployment. This includes monitoring for model drift, conditions for retraining, and conditions for decommissioning.	Explainability  Detail how the model's decisions can be interpreted and understood by stakeholders. Include methods to enhance transparency and communicate decision-making processes effectively.
Objectives  State the specific, measurable goals of the ML project. Detail the expected outcomes and success criteria.		Model Governance  Outline the process for managing model versions including conditions from going from staging to production. Outline procedures for updating and retraining models.			Security  Identify risks related to data breaches, adversarial attacks, and system vulnerabilities. Include measures for safeguarding data and ensuring model robustness against malicious exploitation.

DTU
 By Nicki Skafte Detlefsen nsde@dtu.dk
 From DTU course 02476 Machine Learning Operations
 License: Apache 2.0

[1] https://drive.google.com/file/d/1b9P_WyKMFStKXc2PCGY5E7GsXk3Phl3/view?usp=sharing

Some final loose thoughts...

We don't have the top yet



[1] <https://www.forbes.com/sites/jonmarkman/2026/05/06/anthropic-just-signed-a-compute-deal-with-elon-musks-spacex/>

[2] <https://metr.org/>

Token economics don't hold up in the long run, local rescue?

Julien Chaumond @julien_c · Apr 24
This is where we are right now. And I'm not gonna lie it feels pretty magical 🪄

Qwen3.6 27B running inside of Pi coding agent via Llama.cpp on the MacBook Pro

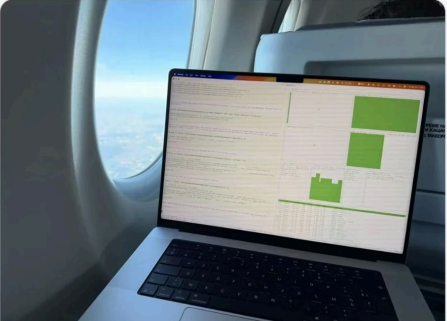
For non-trivial tasks on the @huggingface codebases, this feels very, very close to hitting the latest Opus in Claude Code, or whatever shiny monopolistic closed source API of the day is.

In full airplane mode.

Most people haven't realized this yet.

If you have, it means you have a huge headstart to what I call the second revolution of AI.

Powerful local models for efficiency, security, privacy, sovereignty 🙌

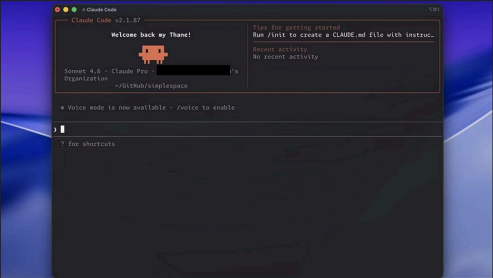


STEALTH SWITCH

Anthropic tested removing Claude Code from the Pro plan

Untenable demand has Anthropic exploring new approaches to rationing its service.

SAMUEL AXON — APR 22, 2026 8:34 PM | 134



→ A screenshot of Claude Code running in the command line. Credit: Samuel Axon

Anthropic caused a stir among developers with what appeared to be a surprise change to

Estimates are that \$200 subscriptions can use \$1000-2000+ compute.

[1] <https://github.blog/news-insights/company-news/github-copilot-is-moving-to-usage-based-billing/>

[2] <https://arstechnica.com/ai/2026/04/anthropic-tested-removing-claude-code-from-the-pro-plan/>

Agentic Engineering is/becomes a real discipline



Judah Diament · 2nd

+ Follow

Dept. Chair & Professor, Computer...
5h · Edited · 🌐

Vibe coding enables people who aren't well trained computer scientists to create complete, working applications. Is this a breakthrough? Not even close - there have been such tools since the late 1980s. See, for example: Apple HyperCard, Sybase PowerBuilder, Borland Delphi, FileMaker, Crystal Reports, Macromedia (and then Adobe) Flash, Microsoft VisualBasic, Rational Rose and other "Model Driven Development" tools, IBM VisualAge, etc. etc. And, of course, they all broke down when anything slightly complicated or unusual needs to be done (as required by every real, financially viable software product or service), just as "vibe coding" does (see <https://lnkd.in/enhAE3Ri>). The only difference is that the outputs of those older tools were actually deterministic and well documented and understood, while your AI prompts and models are not!

To claim that "vibe coding" will replace software engineers, one must: 1) be ignorant of the 40 year history of such tools or 2) have no understanding of how AI works or 3) have no real computer science education and experience or 4) all of the above, OR, most importantly, be someone trying to sell something and make money off of the "vibe coding" fad.

Managers have been vibe coding forever

- tell dev to implement a new feature (vibe coding)
- dev makes changes to code
- manager tests app
- manager does not read the code
- manager complains about bugs
- dev makes changes to fix bugs
- manager doesn't read the code (again)
- dev says "done, try now"
- manager says "gj but be faster next time" or insults the living hell out of the dev
- repeat

NITSN

Recommendations

✔ AI is overhyped. Many people don't know what they're talking about. Especially on LinkedIn, the signal-to-noise ratio is very low. Wait 3-4 months and see what survived.

✔ Unless you're the type who loves to experiment, just pick one model/agent/harness that works for you and stick with that setup.

✔ Control your context before it controls you!

Thank you for your attention 😊

Please feel free to contact me with questions and collaborations

nsde@dtu.dk

skafte nicki@gmail.com

Skill repositories worth checking out:

[1] <https://github.com/anthropics/skills/tree/main/skills>

[2] <https://github.com/mattpocock/skills/tree/main/skills>

[3] <https://github.com/anthropics/knowledge-work-plugins>

[4] <https://github.com/obra/superpowers/tree/main/skills>