Fra idé til prototype

En ~~vibe coding~~ Agentic Engineering workshop del 1

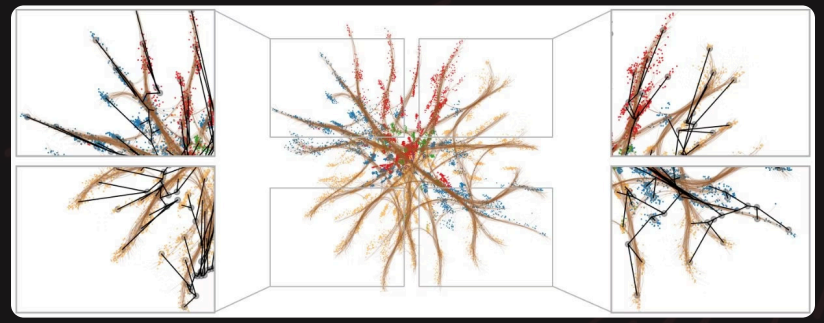
Nicki Skafte Detlefsen, lektor, DTU Compute

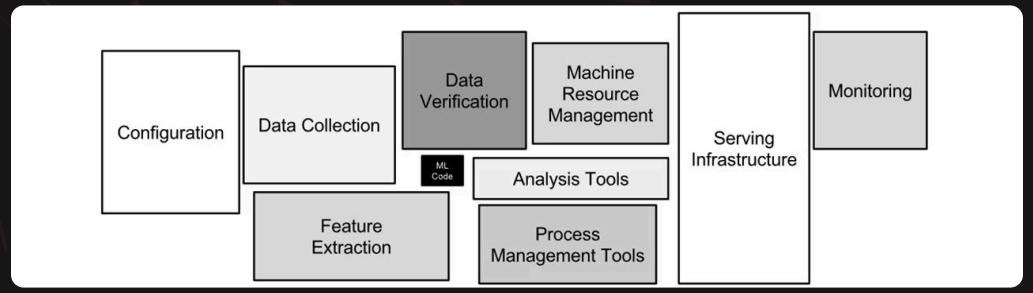
11/05-2026

Hvem er jeg?

💡 Associate Professor på DTU Compute

💡 Senior Forsker ved CAISA
💡 Forskning i MLOps, effektiv machine learning, machine learning development, Agentic Engineering
💡 Open-source developer
💡 Deltids ML Engineer hos **https://lightning.ai/**
💡 Teknisk rådgiver for 5 DTU-startups
= En god blanding mellem den akademiske og industrielle verden





Danmarks vision for AI

Danmark er forpligtet til en fremtid, hvor kunstig intelligens driver udvikling, samtidig med at kerneværdier fastholdes, og både den offentlige og private sektor styrkes.

Borgercentreret AI-udvikling

Udviklingen og anvendelsen af AI skal prioritere borgernes grundlæggende rettigheder og være i overensstemmelse med danske værdier, så den teknologiske udvikling er etisk og ansvarlig.

Global konkurrenceevne for danske virksomheder

Danske virksomheder skal forblive globalt konkurrencedygtige ved at udnytte mulighederne i EU til at udvikle, anvende og sælge AI-løsninger og forretningsmodeller baseret på ansvarlig brug.

Førende offentlig sektor-innovation

Danmark stræber efter at være verdensleder inden for AI i den offentlige sektor og bruge det som et afgørende værktøj til at frigøre arbejdskraft, reducere administration og øge kvaliteten for borgere og virksomheder.

[1] Strategisk indsats for kunstig intelligens, Et styrket fundament for ansvarlig udvikling og anvendelse af kunstig intelligens i Danmark, Danish Goverment, Dec 2024

Suverænitet i AI'ens tidsalder

Suverænitet i AI'ens tidsalder handler ikke om størrelse eller kontrol, men om at kunne vælge sin egen retning og forstå nok til at tage stilling.

AI vil påvirke din virksomhed. Spørgsmålet er, om du former udfaldet — eller bliver formet af andres løsninger, standarder og beslutninger.

For SMB'er handler det om kompetence, mod og viljen til at deltage. Suverænitet kommer af at bruge AI som en styrke, ikke som en sort boks.

Din deltagelse i workshoppen er med til at forme, hvordan AI bruges i virkelige virksomheder med virkelige konsekvenser.

Forord til denne workshop

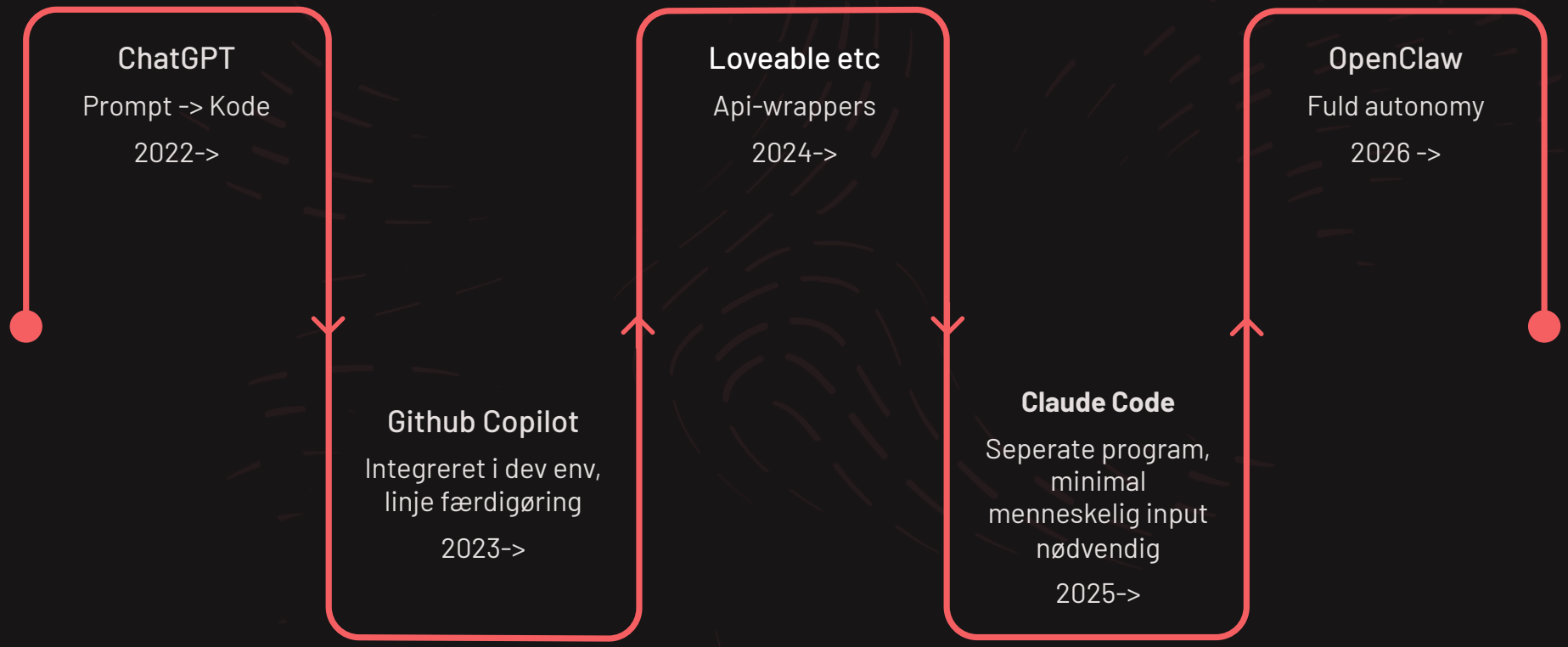
> *Enhver der siger, at de er en agentisk engineering-ekspert, er fuld af bullshit*
>
> Nicki Skafte Detlefsen, ekspert i agentisk engineering

Vi er i fuck around and find out-fasen... så tag derfor alt, der bliver fremført som almen viden, med et gran salt

Vibe coding

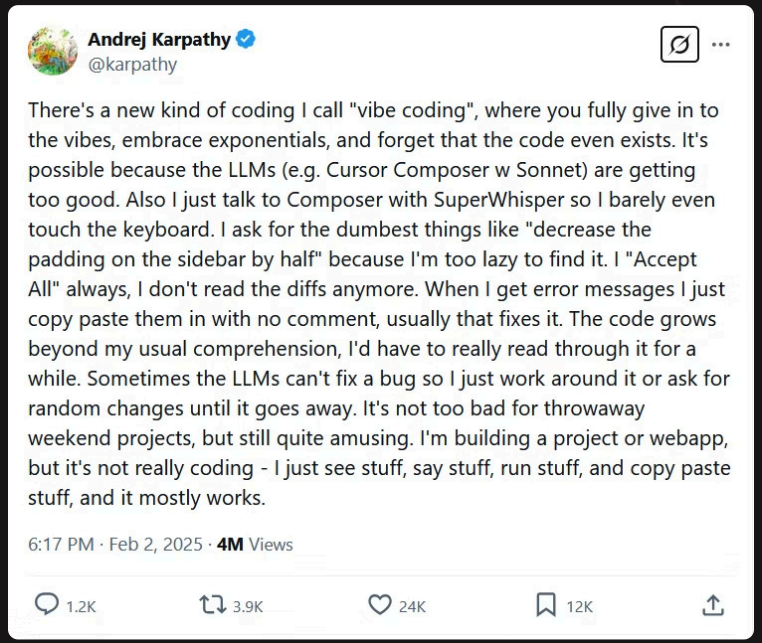
Act 1

Hvordan kom vi hertil



I denne workshop kommer vi til at at beskæftige os med level 3 - Claude code

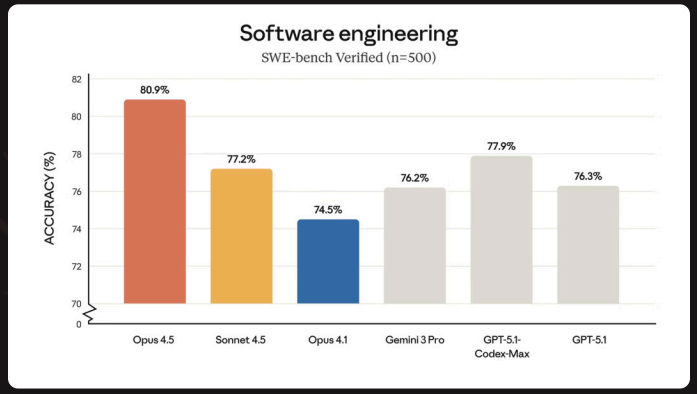
Hvad er Vibe coding?



Andrej Karpathy - OpenAI medstifter - AI-chef hos Tesla

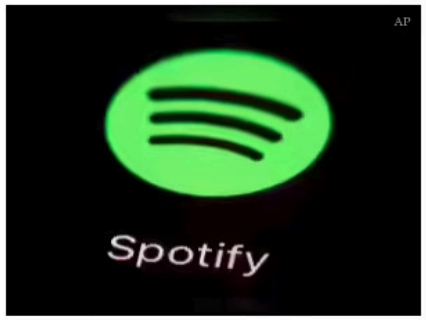
Begrebet refererer til en kodningsmetode, der er afhængig af LLM'er, og som gør det muligt for programmører at generere fungerende kode ved at give naturlige sprogbeskrivelser i stedet for manuelt at skrive i et formelt programmeringssprog.

Bemærk at vibe coding generelt set først har rigtig virket siden opus 4.5 (December 2025)



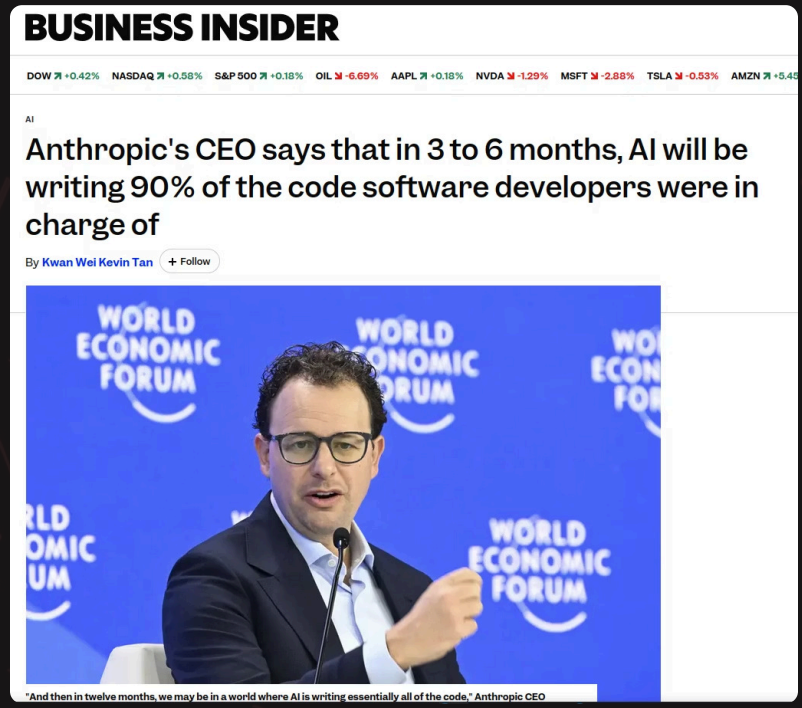
[1] https://www.anthropic.com/news/claude-opus-4-5

Ingen skriver kode længere



Spotify AI coding

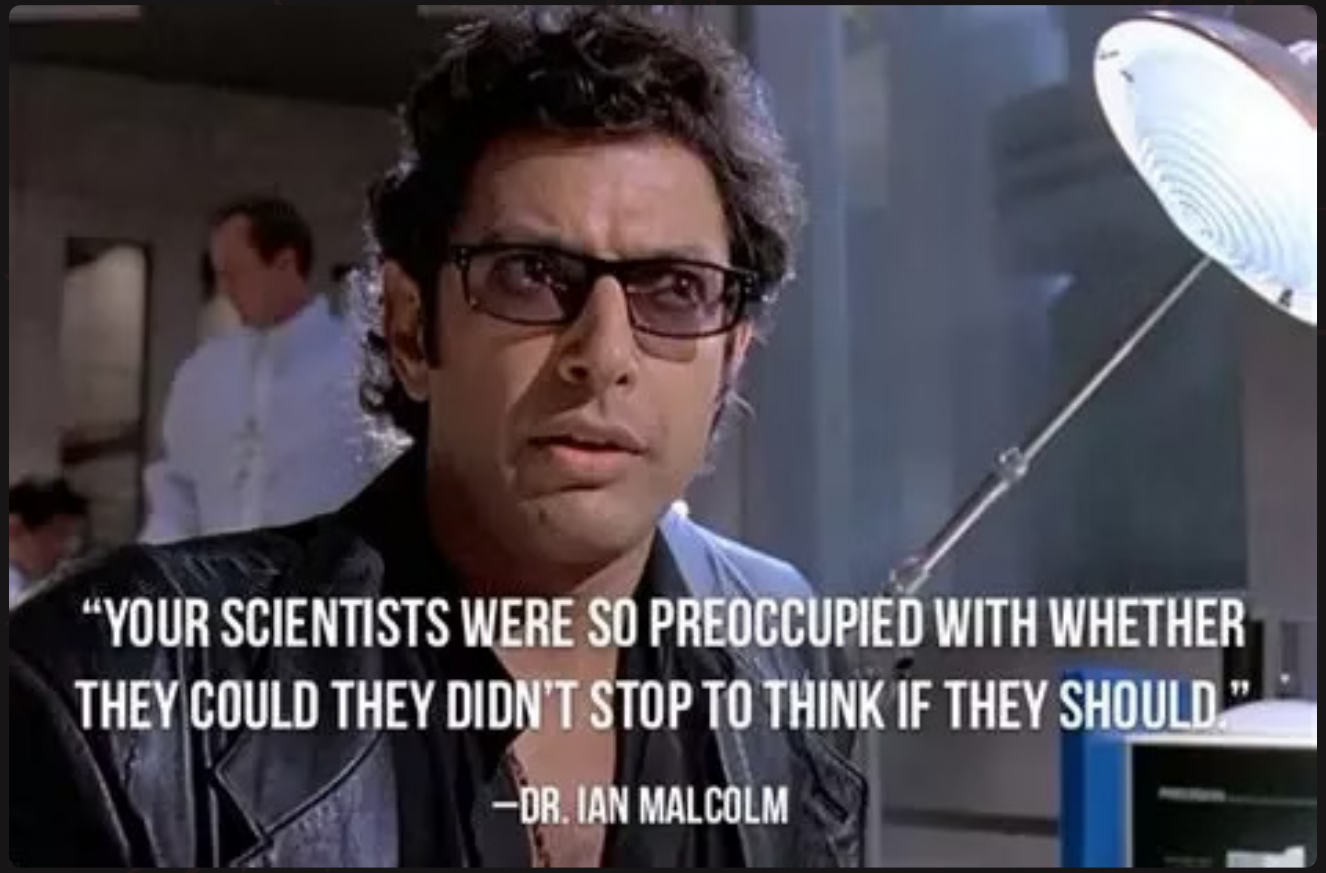
Spotify AI coding: Spotify has reached a new milestone in AI-driven development, according to co-CEO Gustav Söderström, the company's top engineers "have not written a single line of code since December," as quoted in a report. Instead, the work is being handled by the streaming giant's internal AI system,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



[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

Burde vi?



Kan vi finde evidens?

Forbedrer AI-kodning faktisk produktiviteten?

Evidensen siger: Ja — men ikke en "10×" stigning.

Størst forbedring for:

- Generering af boilerplate-kode
- Hjælp til fejlfinding
- Automatiseret dokumentation
- Effektiv API-anvendelse
- **Hurtig prototyping**

Største fordele for:

- Junior- og mellem-niveau udviklere
- Udviklere i ukendte tech stacks
- Opgaver med lille til mellemstort omfang

Vigtige empiriske studier

Studie	Hovedresultat
GitHub Copilot RCT (2022)	Udviklere gennemførte opgaver 55% hurtigere
McKinsey (2023)	GenAI forbedrede arbejdsgange med 20–45%
Microsoft & Stanford studies	Udviklere rapporterer markant højere opfattet produktivitet

Hvorfor “Vibe Coding” også skaber problemer

Hallucinerede API'er & logik

AI kan generere ikke-eksisterende funktioner eller ulogiske kodeforløb.

Sikkerhedssårbarheder

Genereret kode kan indeholde subtile fejl, der skaber sikkerhedsrisici.

Skrøbelige arkitekturer

Overafhængighed af AI kan føre til dårligt strukturerede og svære at vedligeholde systemer.

Reduceret forståelse

Udviklere kan miste dyb indsigt i kodebasen, hvilket hæmmer avanceret fejlfinding og innovation.

“AI Slop” & vedligeholdelsesgæld

Kvaliteten af AI-genereret kode kan være inkonsekvent, hvilket fører til teknisk gæld.

Ydeevnen forringes ved:

- Store, komplekse systemer
- Langsigtet ræsonnering og arkitektoniske beslutninger
- Nye eller ikke-standardiserede problemområder

Vigtig mod-evidens

Studie	Hovedresultat
METR OSS Study (2025)	Erfarne open-source-udviklere blev **19% langsommere** ved brug af AI på komplekse opgaver i den virkelige verden.
SWE-Bench evaluations	Agentiske systemer løser kun et **mindretal** af reelle GitHub-problemer pålideligt.

Konsensus lige nu

AI-kodning fungerer bedst, når:

- → Mennesker forbliver i review-loopet
- → Stærke tests/linting er på plads
- → Agentautonomi er begrænset
- → AI bruges til acceleration, ikke delegering

Nuværende evidens understøtter omtrent 1,2×–2× produktivitetsgevinster i mange arbejdsgange — men ikke pålidelig autonom software engineering.

Men bemærk at evidens fra 2026 er næsten ikke eksisterende.

Derfor...

Konsensus er at værdien af vibe coding meget afhængig af hvem der bruge og hvad det bruges til

Du bliver derfor selv nødt til at prøve det af for at finde ud af om det giver mening for din case.

Lad os sammen få styr på hvordan det virker ✅

Koncepter

Act 2

Lad os forstå det værktøj vi bruger (lidt)

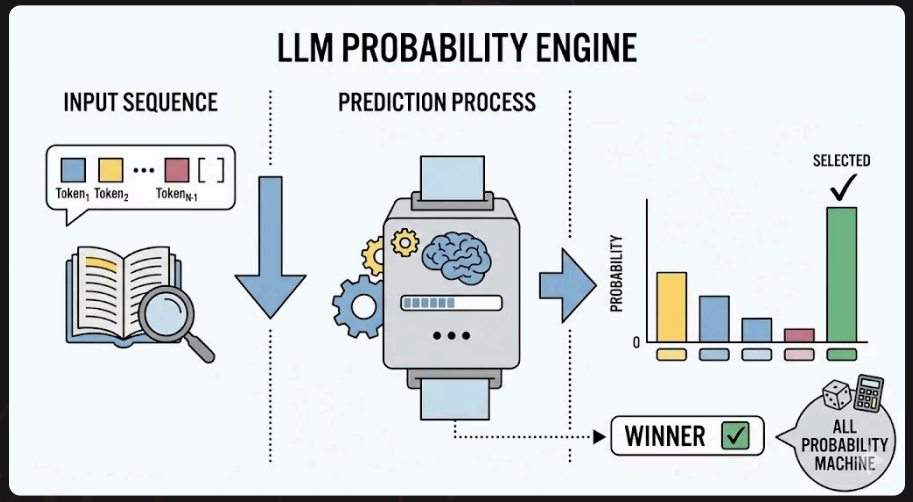
LLM'er er imponerende, men deres grundidé er enkel: de forudsiger det næste token. De "tænker" ikke som mennesker, og derfor er det nyttigt at kende deres styrker og begrænsninger.

Sådan fungerer det

- Den læser din prompt og gætter det mest sandsynlige næste token.
- Den bygger svaret ud fra mønstre i data.
- Den kan lyde sikker, selv når den tager fejl.

Typiske misforståelser

- Den har ikke permanent hukommelse om dig.
- Den slår ikke op i en sandhedsliste.
- Den arbejder kun med den kontekst, du giver den.



Forstå tokens

AI-modeller arbejder ikke direkte med tekst, men med **tokens** – små tekststykker. Det er vigtigt at forstå, fordi tokens styrer både **pris** og **kontekst**.

Hvad er en token?

Et tekststykke på ca. 3-4 tegn på engelsk eller omkring ¾ af et ord. Fx er "Hello world" 2 tokens, mens "Uncharacteristically" er 6.

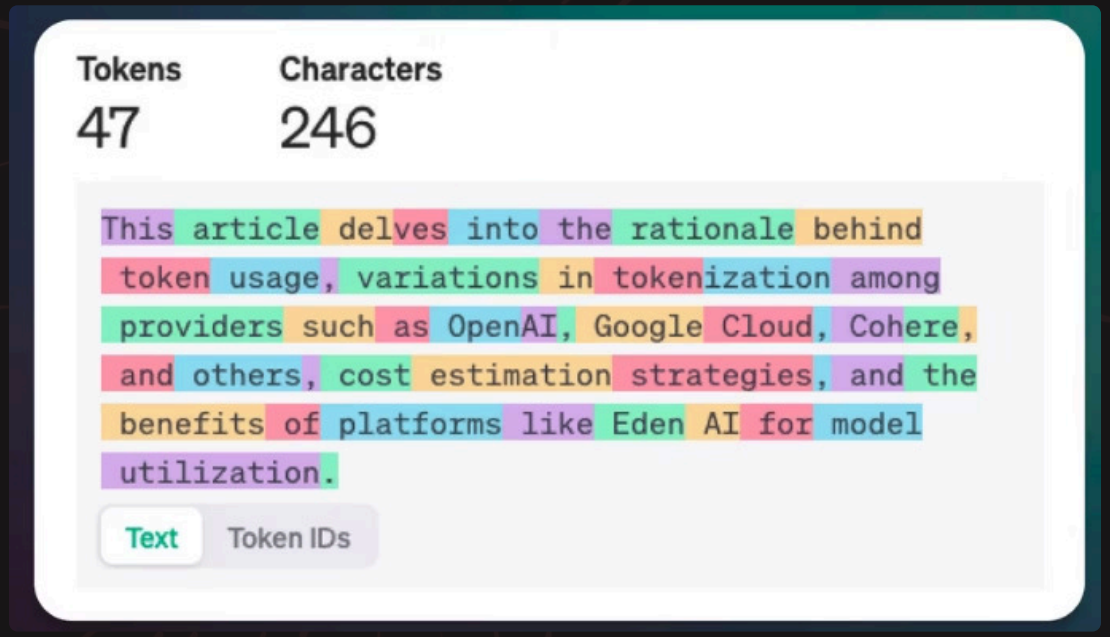
Hvorfor det betyder noget

- Du betaler pr. token for input og output.
- Kontekstvinduer måles i tokens, ikke ord.
- Længere kontekst kan gøre det dyrere og langsommere.

Agentiske loops forstærker forbruget

Når en agent bruger værktøjer i flere trin, vokser tokenforbruget hurtigt med hvert kald, hvert værktøjsoutput og den gemte historik. Derfor er **kontekststyring afgørende**.

Praktiske tommelfingerregler:



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

Prompts

Når AI genererer kode, handler dit fokus mindre om at skrive selv og mere om at styre AI'ens adfærd og kontekst. Du er en "Agent Hyrde", der sætter tydelige rammer for arbejdet.

System Prompts (pr. interaktion)

De styrer den aktuelle samtale og kan fx lyde sådan: *"Du er en senior Python-udvikler. Svar altid med type hints og docstrings. Undgå globale variabler."*

CLAUDE.md / AGENTS.md (projekt-specifik)

Det er din agents permanente system prompt i repoet, med fx **tech stack**, **pytest**-regler og mapper der er **off-limits**.

Uden en `CLAUDE.md/AGENTS.md` fil starter AI'en med nul projektviden, men med den bliver adfærden mere konsekvent og projektspecifik.

Værktøjer: Hvad en agent faktisk kan gøre

En LLM uden værktøjer er som en meget klog person låst inde i et rum med ingen forbindelse til omverdenen.

Et værktøj er en funktion, som agenten kan vælge at kalde, observere resultatet af og ræsonnere over, før den tager det næste skridt.

Websøgning

Læs/Skriv fil

Kør kode

API-kald

Send e-mail

Agenten beslutter selv, hvornår og om den skal kalde et værktøj – det er ikke deterministisk.

Kvaliteten af din agent er i høj grad begrænset af kvaliteten af dens værktøjer.

MCP — Model Context Protocol

MCP er en åben standard, der lader agenter opdage og kalde værktøjer på tværs af systemer — en slags USB-C til AI-integrationer.

Før MCP

Hver værktøjsintegration var skræddersyet og skrøbelig, hvilket gjorde systemer svære at vedligeholde og skalere.

Med MCP

Standardiserer, hvordan værktøjer beskrives og kaldes, hvilket skaber en mere robust og interoperabel tilgang.

Opdagelig

Agenten kan dynamisk finde og forstå tilgængelige værktøjer under kørsel uden forudgående konfiguration.

Sammensætbar

Gør det muligt at blande værktøjer fra forskellige servere og tjenester for at opbygge komplekse agenter.

Standardiseret

Sikrer, at den samme protokol anvendes uanset værktøjets udbyder, hvilket forenkler integration.

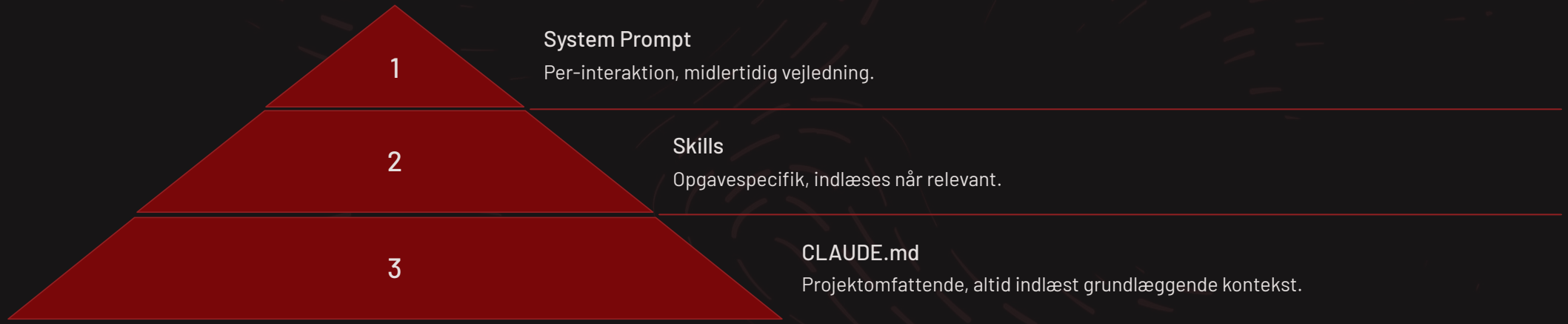
Hver MCP-værktøjsdefinition forbruger tokens i kontekstvinduet. At tilføje 10 værktøjer fra en server medfører et overhead, selvom de ikke kaldes.

⚠ Tilføj kun de MCP-servere, du faktisk bruger — "tool bloat" er en stille kontekstdræber.

Skills

En skill er en genanvendelig instruktionsfil, der hjælper agenten med én bestemt opgave. Tænk på det sådan: hvis din `CLAUDE.md`-fil er medarbejderhåndbogen, så er skills SOPs (Standard Operating Procedure) for specifikke opgaver.

Agentens hierarkiske instruktionssæt



Skills defineres ofte som markdown-filer som `code-review.md`, `write-commit-message.md`, `summarize-document.md` og `security-check.md`.

Vigtige fordele ved modulære skills

- Skills kan **versioneres**, hvilket muliggør kontrollerede opdateringer og tilbagerulninger.
- De er **testbare**, hvilket sikrer pålidelig performance til specifikke opgaver.
- Skills kan **deles på tværs af projekter**, hvilket fremmer genanvendelighed og konsistens.

Lidt mere praktik

Act 3

LLM vs Agent vs Agent-harness

Koncept	Hvad det er	Nøgleegenskaber
LLM-model	Et neuralt netværk, der forudsiger tokens baseret på input.	Stateless, udfører en enkelt forward pass og kan ikke udføre handlinger eller interagere med eksterne systemer.
Agent	En LLM udstyret med værktøjer og en ræsonneringsløkke.	Kan handle på information, observere resultater og iterere mod et mål ved at lære af interaktioner.
Agent-harness	Den understøttende ramme, der orkestrerer én eller flere agenter.	Håndterer tilstand, routing, retries, guardrails og giver sikker adgang til forskellige værktøjer og eksterne datakilder.

Agent + Agent-harness går hånd i hånd

Et praktisk eksempel: Opsummering af regnskab

Lad os se et konkret eksempel på, hvordan samspillet mellem et Agent-harness, en agent og en LLM udfolder sig i praksis for at løse en opgave.

Brugerforespørgsel

Du beder dit Agent-harness om: **"Find de seneste kvartalsregnskaber for [Virksomhed X] og giv mig en opsummering."**

Harness'en orkestrerer

Harness'en modtager din forespørgsel og initialiserer en passende agent (baseret på dens LLM) til opgaven. Den sikrer også, at agenten har adgang til de nødvendige værktøjer.

Agenten analyserer & handler

Agenten (LLM) ræsonnerer: "Jeg skal finde information online." Den vælger at bruge et **"websøgning"-værktøj**, søger efter "[Virksomhed X] kvartalsregnskaber" og finder relevante links.

Agenten eksekverer & syntetiserer

Agenten (LLM) læser indholdet af den identificerede regnskabsrapport (via et **"læs dokument"-værktøj**), udtrækker nøgletal og centrale pointer om performance.

Resultat leveres

Den opsummerede earningsrapport sendes tilbage fra agenten til harness'en, som præsenterer den til dig på en letforståelig måde. Hele processen er en agentisk løkke.

Dette iterative samspil mellem modellens intelligens, værktøjernes funktionalitet og harness'ens styring er kernen i avancerede AI-applikationer.

Valg af den rigtige model (Antropic som eksempel)

At vælge den optimale AI-model er afgørende for at balancere ydeevne, hastighed og omkostninger.

At forstå styrkerne ved hver model hjælper med at træffe informerede beslutninger for dine projekter.

Model	Bedst til	Hastighed	Omkostning (pr. MTok)	Kontekstvindue
Haiku v4.5	Klassificering, routing, opsummering, højvolumenopgaver	Hurtigst	$1 / $5	200K
Sonnet v4.6	De fleste produktionsarbejdsopgaver — kodning, analyse, RAG	~2× langsommere end Haiku	$3 / $15	1M
Opus v4.7	Kompleks ræsonnering, dybe agentiske opgaver, frontier-kodning	Langsomst	$5 / $25	1M

Beslutningsheuristikker

Vælg som standard Sonnet

Den håndterer 90%+ af opgaverne til en brøkdel af Opus' pris, hvilket gør den til arbejdshesten for de fleste produktionsscenarier.

Brug Haiku som router

Brug Haiku som et første filter i multi-model pipelines eller til at klassificere intention og sende komplekse sager videre til Sonnet eller Opus for at reducere omkostningerne markant i højvolumen-systemer.

Reserver Opus til kritiske opgaver

Vælg kun Opus, når opgaven reelt kræver dens avancerede muligheder, såsom langvarigt agentisk arbejde eller kompleks flertrinsræsonnering.

Vælg den rigtige agent

Svaret er som ofte i dag at man har flere agenter som alle har:

- Forskellige system prompts e.g. roller
- Forskellige adgang til tools
- Forskellige

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 ---

Harness engineering

Orkestrering - Styrer agenternes flow.

Tilstandsstyring - Bevarer kontekst mellem sessioner.

Routing - Sender opgaver til den rette agent.

Guardrails - Sætter klare sikkerhedsgrænser.

Fejltolerance - Håndterer fejl og genforsøg robust.

Sikker adgang - Kontrollerer adgang til systemer og data.

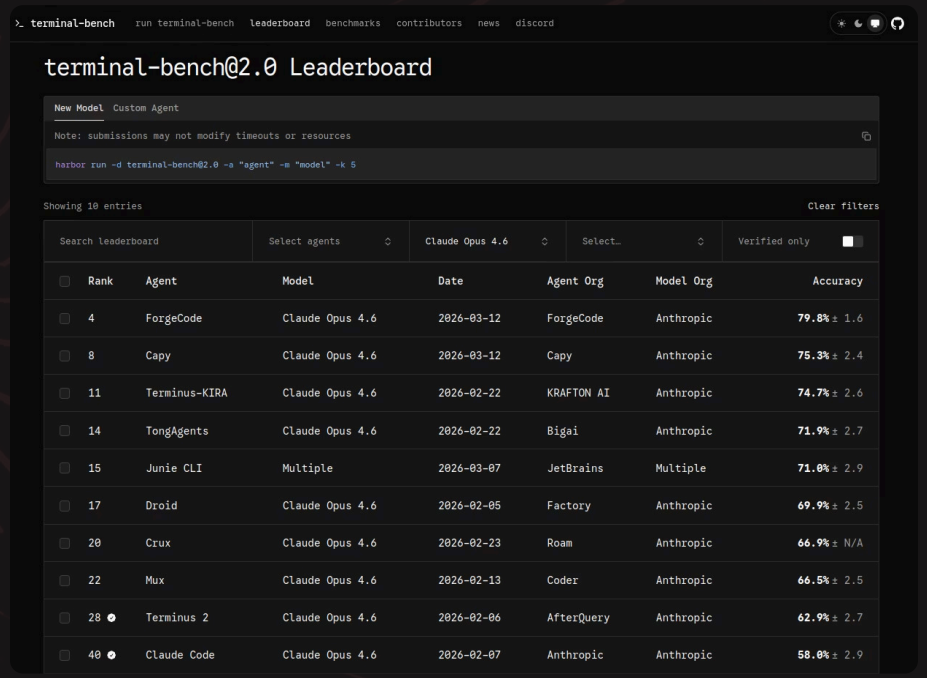
Closed Source

- Claude Code
- Codex
- Antigravity
- Cursor

Open Source

- Pi
- Opencode
- OpenHarness

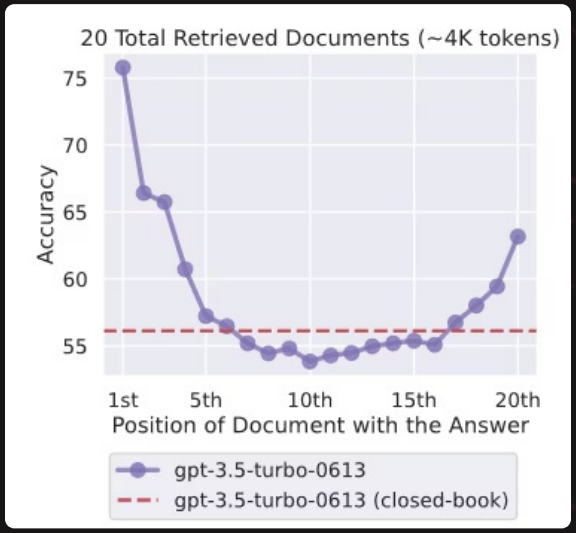
Hvorfor har valg af Harness betydning?



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

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

Smart/dum-zone

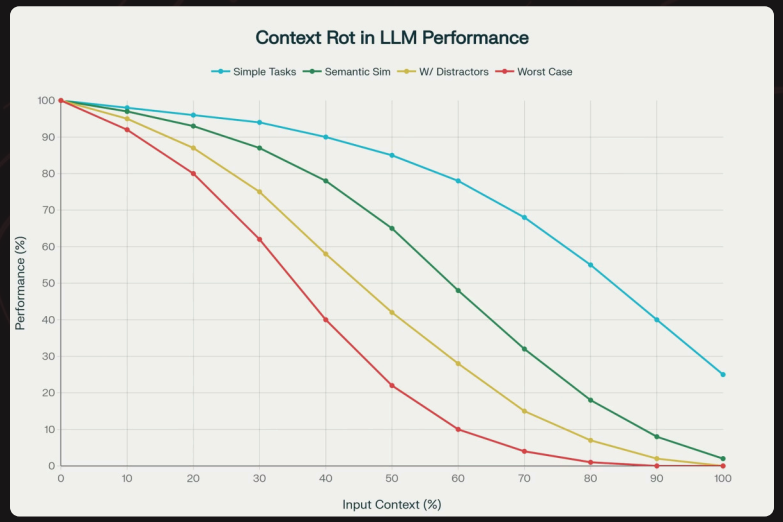


Agenter lider under "lost in the middle" problem:

Alt information der er i midten af context vinduet bliver glemt

Hver token, der tilføjes til context-vinduet, øger den beregningsmæssige kompleksitet kvadratisk – fordi attention skal forbinde hver token med alle andre tokens. Det svarer til at tilføje et hold til en fodboldliga: antallet af kampe vokser kvadratisk.

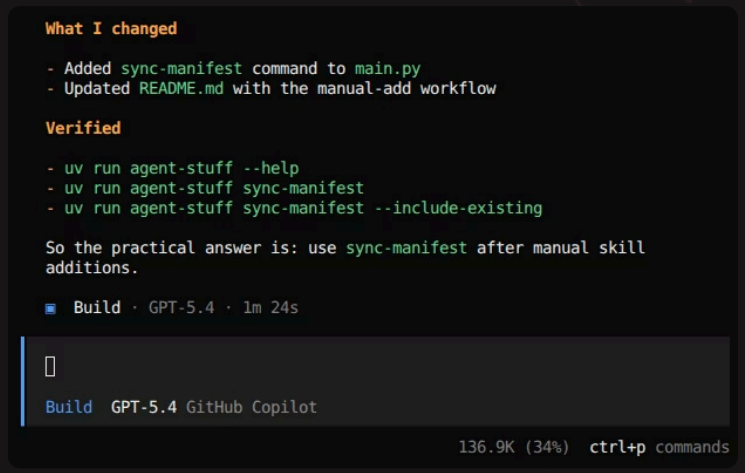
Omkring ~100k tokens begynder modellen at blive "dummere" – også selvom den har et context window på 200k eller 1M tokens – og den træffer ofte dårligere beslutninger.



Det praktiske takeaway: hold context windows korte, og start nye samtaler ofte.

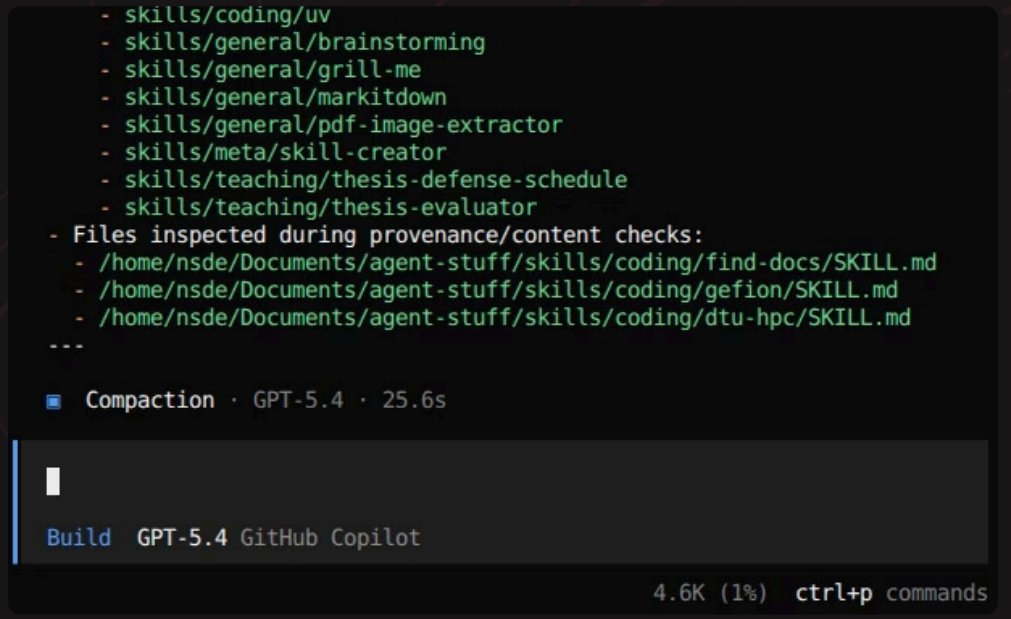
[1] https://arxiv.org/pdf/2307.03172

Hvordan holder du styr



Context tællingen er et af de vigtigste feedback signaler du har, både til at holde prisen nede og holde din model smart.

bruge /compact til at holde samtalen kørende og bringe din model tilbage til smart-zone context length.



Men er prompt engineering ikke fint nok?

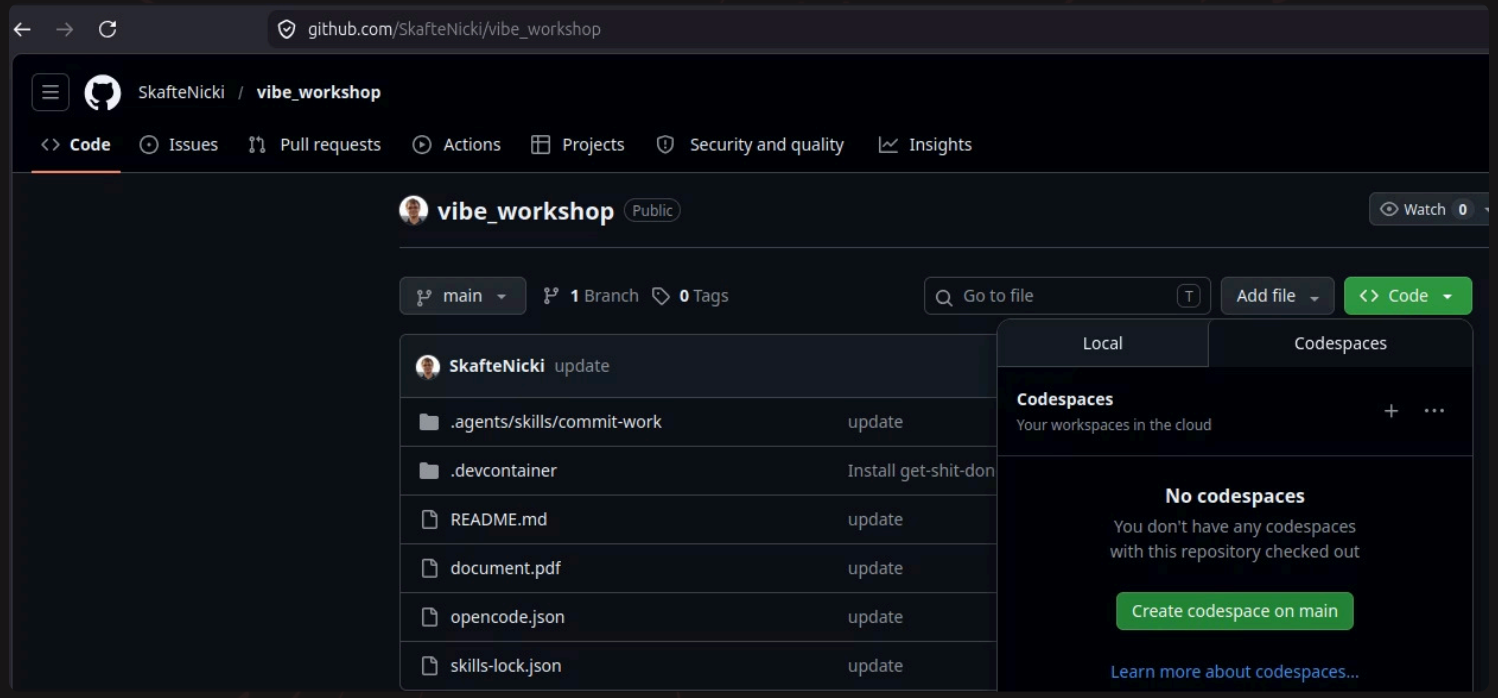
Dimension	Prompt Engineering	Context Engineering
Fokusområde	Optimering af individuelle instruktioner	Omfattende informationsøkosystem
Omfang	Ord, formuleringer, eksempler	Værktøjer, hukommelse, dataarkitektur, struktur
Vedvarende effekt	Stateless, ingen hukommelseslagring	Stateful med langtidshukommelse
Skalerbarhed	Begrænset og skrøbelig i stor skala	Meget skalerbar og robust
Bedst til	Engangsopgaver, indholdsgenerering	AI-applikationer i produktionskvalitet
Kompleksitet	Lav adgangsbarriere	Høj, kræver ekspertise i systemdesign
Pålidelighed	Uforudsigelig i stor skala	Konsekvent og pålidelig
Vedligeholdelse	Skrøbelig over for ændringer i krav	Modulær og vedligeholdelsesvenlig

Til tastaturet!

Act 4

Opgave til jer

1. Hvis du ikke har en, lav en gratis Github konto **https://github.com/signup**
2. Når du er logget ind, gå til **https://github.com/SkafteNicki/vibe_workshop**
3. Lav et gratis codespace
4. Når codespace er færdig med at blive initialiseret, så åben **README.md** filen (click) og følg instruktioner



API-key: **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**

Opgaver at vælge i mellem

Let

`exercises/01-create-and-improve-a-skill.md`

Lav en skill der omdanner mødenoter til handlingspunkter, test og forbedr den derefter.

Mellem

`exercises/03-plan-prd-build.md`

Læs et produktcase, skriv en plan, og byg en lille prototype.

Svær

`exercises/04-your-own-idea.md`

Bring din egen idé, form den med AI, stresstest den, og byg den.

Kan vi se et par demoer?

https://bit.ly/48NNw83

(should open a teams meeting)

Afslutning

Act 5

Hvorfor dette er en ny færdighed, ikke bare et nyt værktøj

Arkitekten & anmelderen

Du går fra at kode til at designe og kvalitetssikre AI-løsninger. Det vigtigste bliver at formulere krav præcist.

Fra at skrive til at dirigere

Arbejdet handler mere om at styre AI-agenter end om linje-for-linje-kode. Du guider processen i stedet for at bygge alt selv.

Funktionelle prototyper, hurtigt

Du kan skabe imponerende prototyper på en dag. Ikke færdig software, men noget der hurtigt gør idéer konkrete.

Anbefalinger

- AI er overhypet. Der er mange som ikke ved hvad de snakker om. Især på Linkedin er signal til støj meget lav.

  Vent 3-4 måneder og se, hvad der overlevede.

- Medmindre du er typen, der elsker at eksperimentere, så vælg bare én model/agent/harness der fungere for dig og hold fast i den opsætning

- Du kan ikke have for mange skills. Fight me.

- Aldrig bare kopier en instruktion/færdighed/agentdefinition fra internettet uden at have læst, hvad den gør/indeholder.

- Hold mængde af context under 100.000. Din agent bliver kun dummer derfra.

Slutningen

Du har nu en prototype.

Workshop 2: Fra prototype til produkt

> *Den brutale realitet — hvad din prototype ikke har, og hvordan man lukker det gab.*

✅ Hvad der mangler

✅ MLOps & EvalOps — gør AI-systemer pålidelige

✅ Sikkerhed — hvad der går galt, når virkelige brugere dukker op

✅ Vejen fra "det virker på min maskine" til "det virker"

Tak for jeres opmærksomhed 😀

Kontakt mig gerne med spørgsmål og samarbejder

nsde@dtu.dk

skaftenicki@gmail.com