Welcome to Agent skills workshop

Goal of the workshop

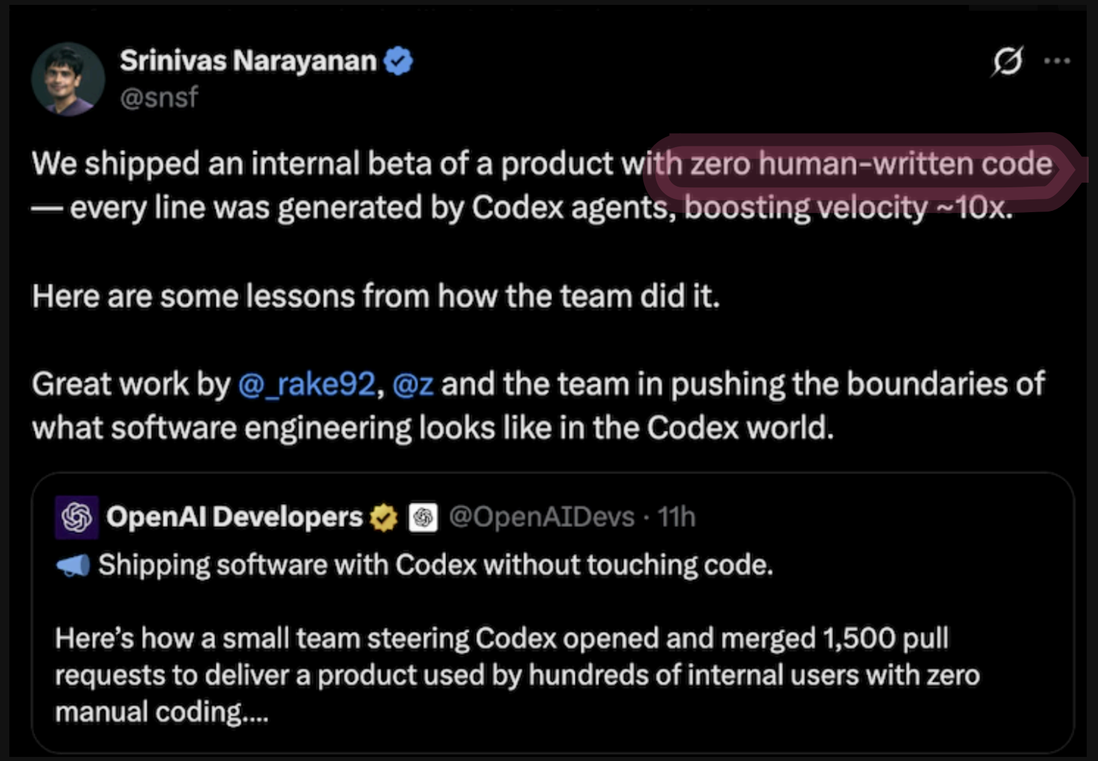
1. Basics of agents
2. Use persistent agents parallelly (my workflow)
3. Learn to use skills and write custom skills
4. Learn to collaborate as a team

Why do we need to use agents ?

Boris Cherry, an engineer anthropic, has publicly stated that Claude code has written 100% of his contributions to Claud code. Not “majority” not he has to fix a “couple of lines.” He said 100%.

→ He hasn't written a single line of code manually since NOV 2025 but ships several PRs per day

Why do we need to use agents ?



Srinivas Narayanan
@snsf

We shipped an internal beta of a product with zero human-written code — every line was generated by Codex agents, boosting velocity ~10x.

Here are some lessons from how the team did it.

Great work by @_rake92, @z and the team in pushing the boundaries of what software engineering looks like in the Codex world.

> **OpenAI Developers** @OpenAIDevs · 11h
> Shipping software with Codex without touching code.
>
> Here's how a small team steering Codex opened and merged 1,500 pull requests to deliver a product used by hundreds of internal users with zero manual coding....

→ zero human-written code is the new standard

February 11, 2026 Engineering

Harness engineering: leveraging Codex in an agent-first world

By Ryan Lopopolo, Member of the Technical Staff

Over the past five months, our team has been running an experiment: building and shipping an internal beta of a software product with **0 lines of manually-written code**.

The product has internal daily users and external alpha testers. It ships, deploys, breaks, and gets fixed. What's different is that every line of code—application logic, tests, CI configuration, documentation, observability, and internal tooling—has been written by Codex. We estimate that we built this in about 1/10th the time it would have taken to write the code by hand.

question on vel

Why do we need to use agents ?

Andrej Karpathy @karpathy

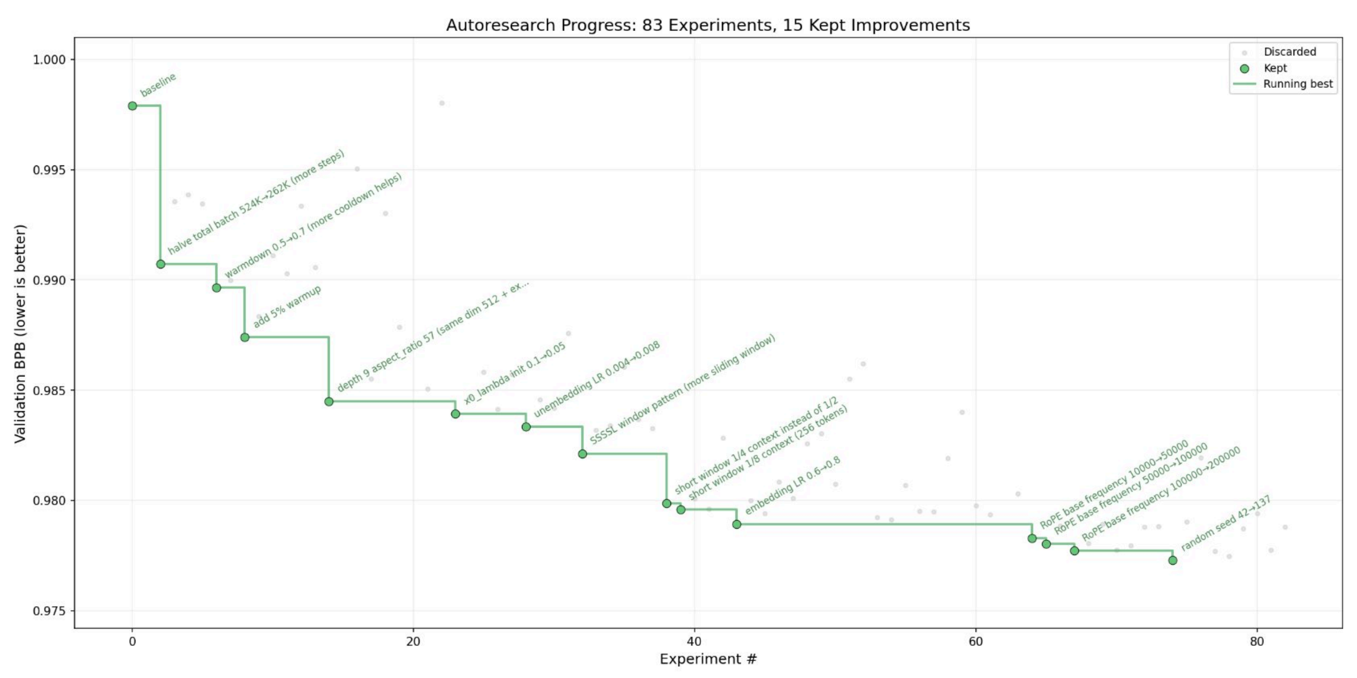
I packaged up the "autoresearch" project into a new self-contained minimal repo if people would like to play over the weekend. It's basically nanochat LLM training core stripped down to a single-GPU, one file version of ~630 lines of code, then:

- the human iterates on the prompt (.md)
- the AI agent iterates on the training code (.py)

The goal is to engineer your agents to make the fastest research progress indefinitely and without any of your own involvement. In the image, every dot is a complete LLM training run that lasts exactly 5 minutes. The agent works in an autonomous loop on a git feature branch and accumulates git commits to the training script as it finds better settings (of lower validation loss by the end) of the neural network architecture, the optimizer, all the hyperparameters, etc. You can imagine comparing the research progress of different prompts, different agents, etc.

→ Research is also being automated with agents

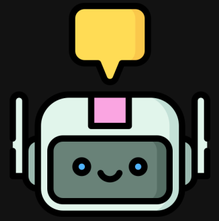
autoresearch



One day, frontier AI research used to be done by meat computers in between eating, sleeping, having other fun, and synchronizing once in a while using sound wave interconnect in the ritual of "group meeting". That era is long gone. Research is now entirely the domain of autonomous swarms of AI agents running across compute cluster megastructures in the skies. The agents claim that we are now in the 10,205th generation of the code base, in any case no one could tell if that's right or wrong as the "code" is now a self-modifying binary that has grown beyond human comprehension. This repo is the story of how it all began. -@karpathy, March 2026.

question on vel

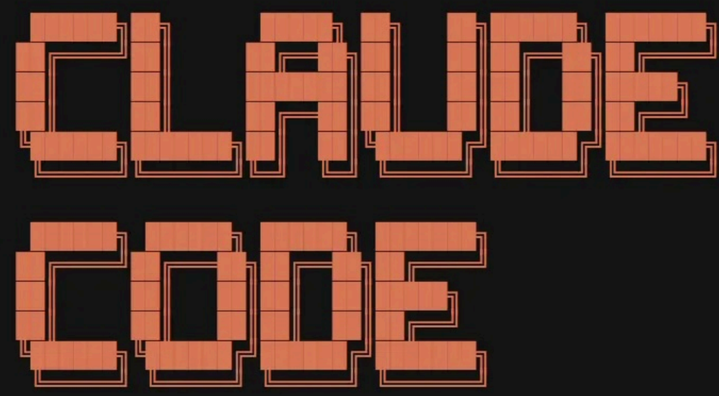
Practical guidelines to using agents



1. As a researcher
   - My personal story
   - using agents to write complete code
   - running agents in parallel
   - using agent skills
2. As a team
   - share custom instructions / Agents md
   - share skills
   - share knowledge

My personal story

I Started using claude code in Oct 2025



It was powerful but ....

- Since I was paying with my own money, I was often ratelimited
- Its CLI based, so I felt it was harder to control and see the changes it was doing
- Overall, good experience but I feel its not that good compared to todays SOTA agents (specially with Opus 4.6)

My personal story

Now:

- I run almost 5 agent sessions a day (parallelly all the time)

- I am able to achieve in a day what would other wise take me a week or so if I do it manually without agents

- Needed some getting used

    - tragically different from traditional software eng.
    (write code - test -> write prompt - validate)

    - Felt out of control initially :(

Takeaway:
Try to resist the urge to write the code urself

Takeaway:
let agents take control

Writing complete code

Before :

```
python3 -m src.inference.infer_mm\
    --base_checkpoint /data/care/checkpoints/phi4-stage-2-with-clf-head-NH-specific/checkpoint-99258\
    --checkpoint /data/care/output_checkpoints/grpo-grounding-1-v1-20260304-121533/checkpoint-3500\
    --output /data/care/grpo/outputs/grpo-grounding-1-v1-20260304-121533/test_abnormality_grounding_padchest.jsonl\
    --json_path "/data/care/CARE_DATASETS/version_4_with_single_prompt_variations/grounding/test/test_abnormality_grounding_padchest.json"\
    --img_root "/data/care/interpret-cxr-challenge/data"
```

commands like this were used to run inference

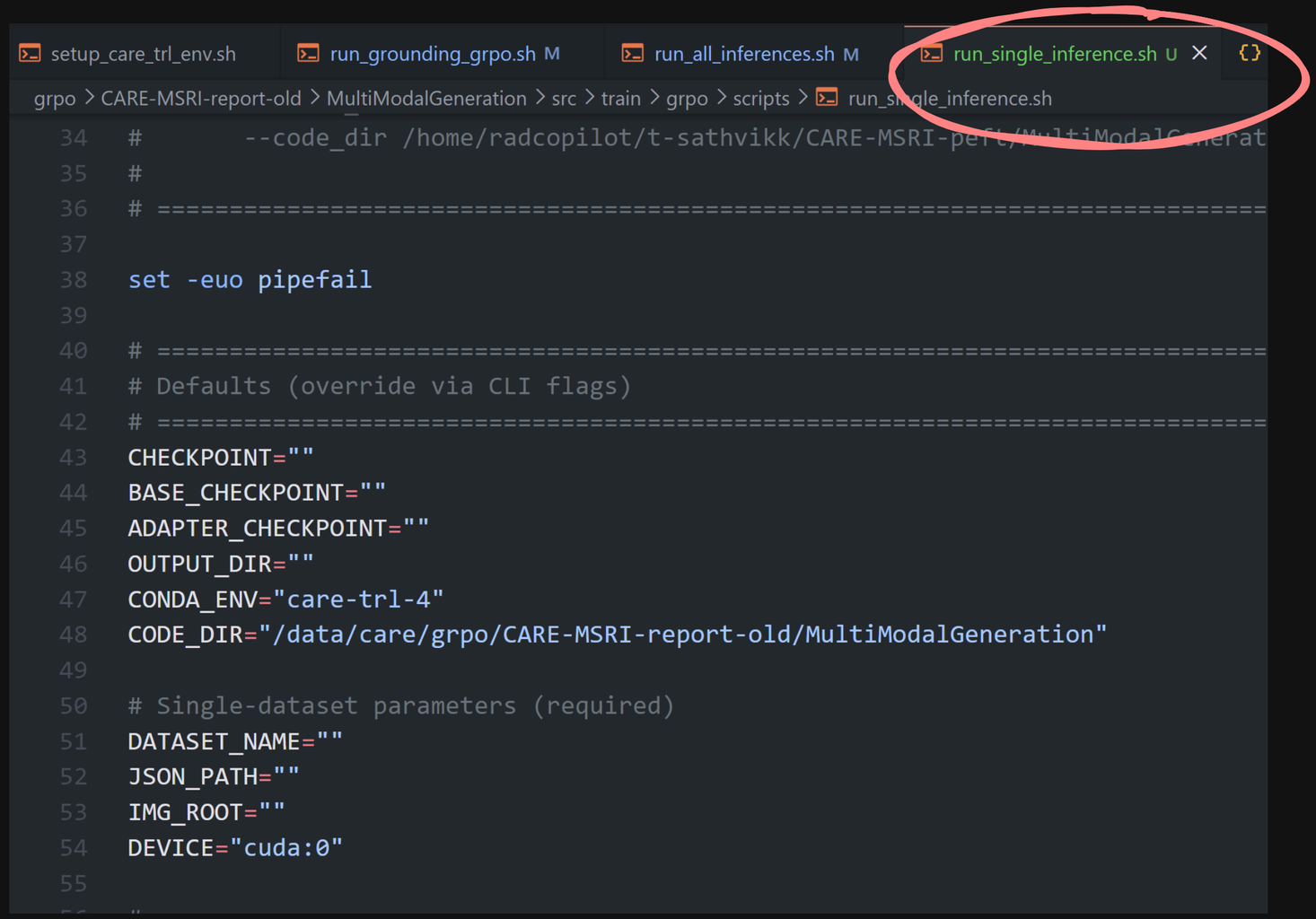
problems:
- you need to remember the arguments
- not easy to change the arguments in cli

Writing complete code

After:

- Asked the agent to write script

Pros:
- Easy to edit
- Easy to integrate to skill (later)
- Easy for others to run
- Easy to document
- Easy to document the guide to run this script



Writing complete code

Learning

- Use agents to write more accessible code

- Focus your efforts on documentation of your methods/ knowledge / workflows

- Write more general scripts not just for the task at hand

question on vel

Running agents in parallel

Answer the question in the quiz

Practical hands on show !! (Yaay)

(Open your terminal guys lets get hands on)

cover yolo, chat and cli, tmux adv commands, worktress, cli commands

Running agents in parallel

Important things to consider while running persistent agents

- Keep one focused agent per task (some rules to manage agents, book keeping, short lived agents)

Takeaways:

As the no of agents grows, remembering what agents are running and for what becomes difficult, so name them properly
use tmux naming, simple note book keeping also works

Running agents in parallel

Important things to consider while running persistent agents

- Use git worktrees
- Use one git worktree per task
(Eg: one for grounding task)

- Merge with main as and when done with task

- A healthy git usage will save you a lot of effort

question on skill

Running agents in parallel

Important things to consider while running persistent agents

- Use git worktrees
- Use one git worktree per task
(Eg: one for grounding task)

- Merge with main as and when done with task

- A healthy git usage will save you a lot of effort

question on skill

Skills

on-demand domain knowledge that can be loaded when needed — giving the agent the right expertise at the right time, without keeping all that knowledge in memory constantly.

- agentskills.io is the specification hub — it defines the rules for how SKILL.md files

show
- agentskills.io, vs code agent skills

question on vel

show
- in terminal

Skills

Where does the agent look for skills ??

VS Code supports two types of skill locations

-- project skills stored in your repository
(.github/skills/, .claude/skills/, .agents/skills/)

useful for team collobration

-- personal skills stored in your user profile
(~/.copilot/skills/, ~/.claude/skills/, ~/.agents/skills/).

-- add skills with `/skills add` in the cli

Skills

Practical example of rexrank eval skill

- Agent cant be alive for more than 5-10 mins
  - Check and then return to user

- maintaining eval_state.json if context switch happens

- resources in a skill are also imp (ex: filetransfer skill)

show
- in terminal

Skills

Practical example of rexrank eval skill

- Keep iterating skills (they are not meant to be static)

_ Make it more general

- Your skills will compound over time

- Make the agent itself iterate on skills over time

custom instructions

Custom instructions are markdown files that automatically inject project-specific rules and conventions
into every Copilot chat request, so the AI follows your codebase's patterns without you repeating them in
every prompt

```
.github/
├── copilot-instructions.md
└── instructions/
    └── grpo.instructions.md
```

show
- in terminal

custom instructions

- Dont clutter the main copilt-instructions
- Make use of `applyTo` parameter
- Keep updating the instructions.md file regularly

How to collaborate as a team

- Write skills and share them with others

- Maintain good instructions files

- Maintain good docs file (show the blog post)

> As a team we have to put effort once and reap the fruits forever !