

OpenAI

Codex 201 - Advanced Agent Setup and Repeatable Workflows

How to get more out of Codex



Name goes here

Title goes here



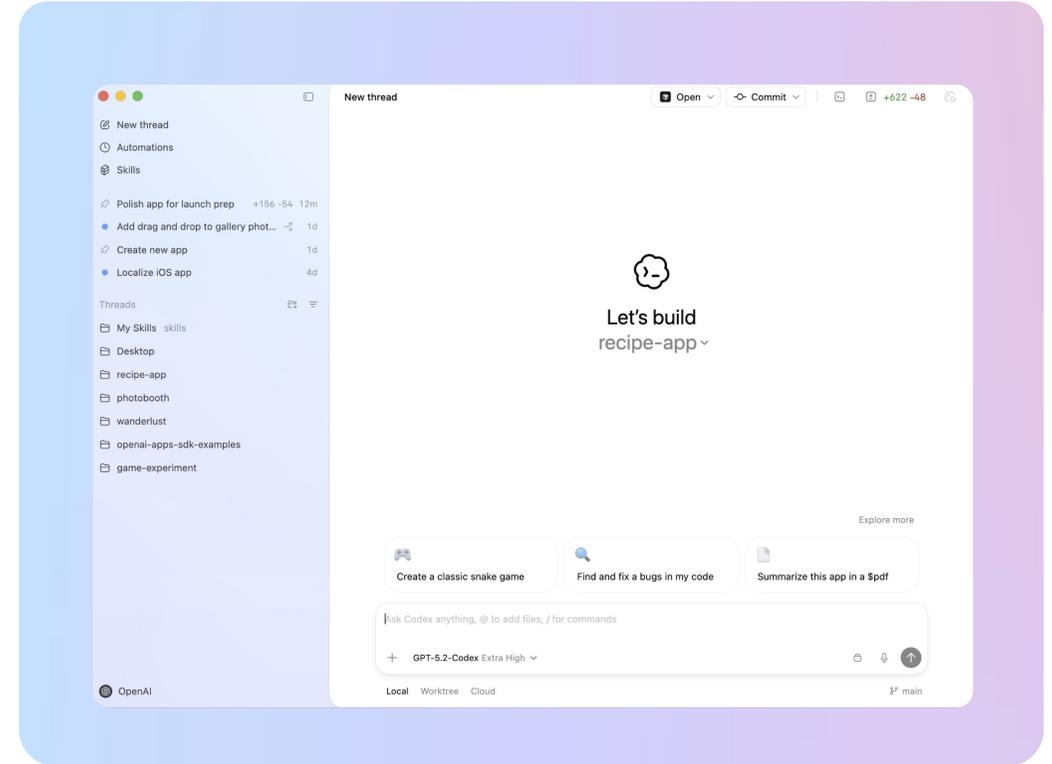
- 01 Codex Intro
- 02 Configurations
- 03 MCPs
- 04 Skills
- 05 Automations

What have you been using
Codex for? Anything to
highlight for the group?

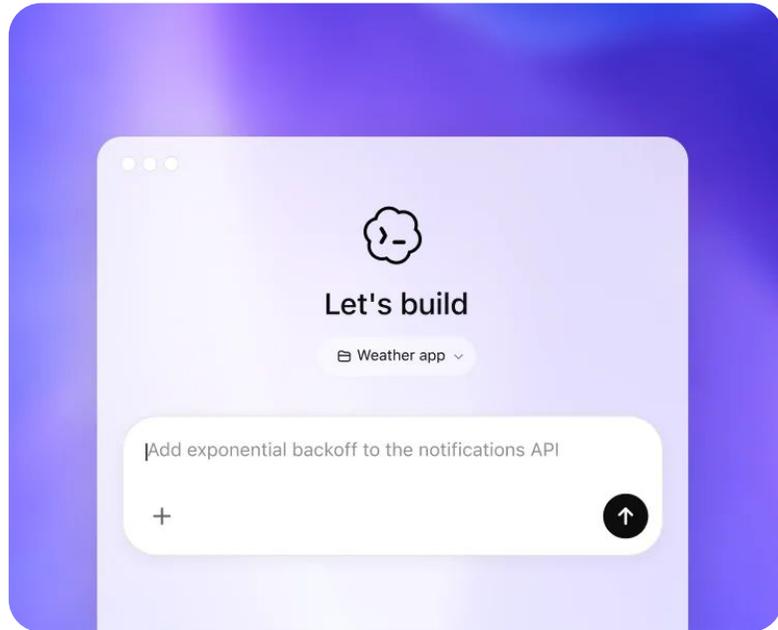
How are you using Codex?



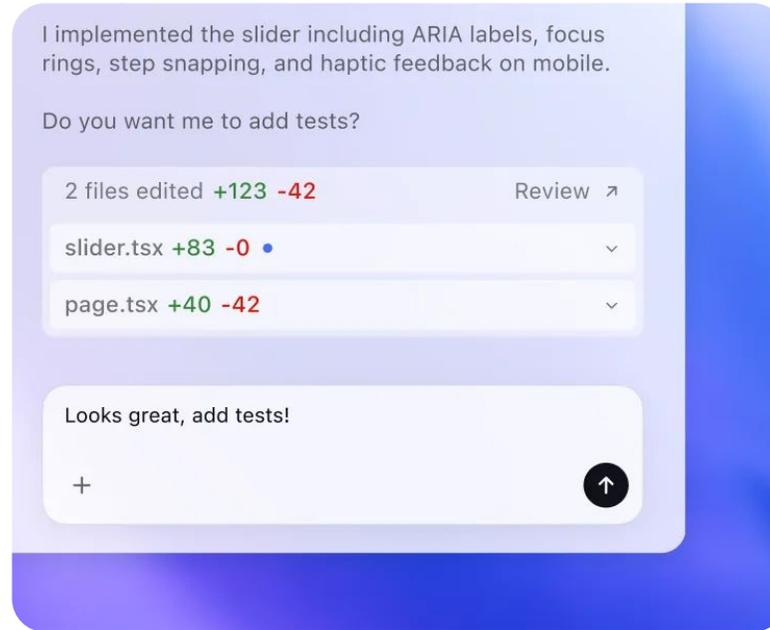
OR



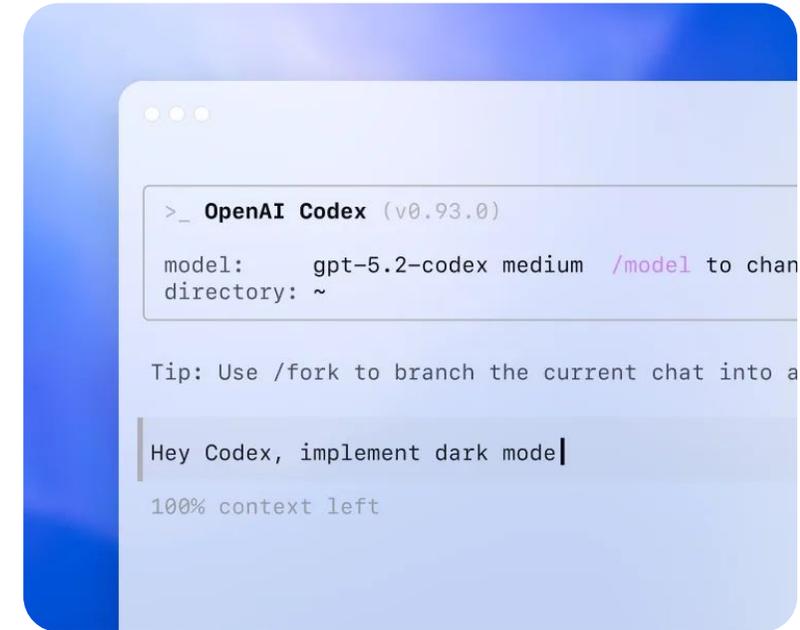
The same agent everywhere you build



Start in the app



Continue in your IDE



Keep going in the terminal



To recap how to
delegate
to coding agents

- [] Shared understanding and context
- [] Boundaries and expectations
- [] Collaborative planning
 - ↳ steps
 - ↳ interfaces
 - ↳ acceptance criteria
 - ↳ key tests and documentation
 - ↳ rollout plan

A new operating model for engineering

01

Developers

02

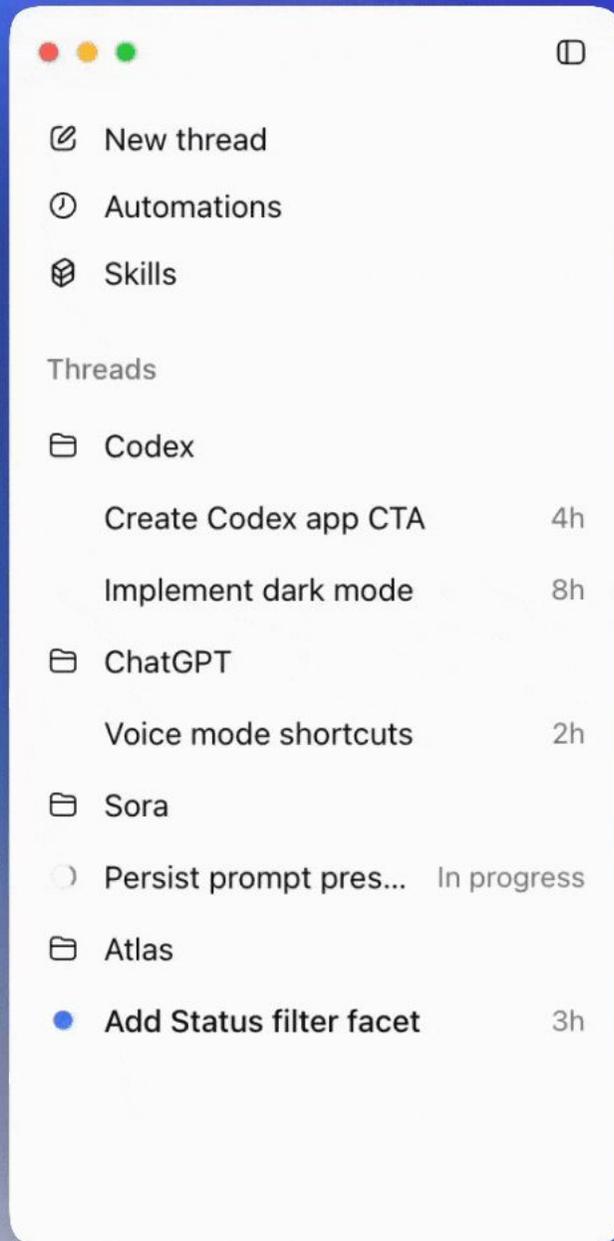
Teams

03

Organizations

Turn context into team defaults

- 01 **Standardize behavior with configurations**
Set defaults once: approvals, sandboxing, and which tools are enabled so results are consistent across developers and environments
- 02 **Connect to your tools with MCPs**
Stop copy pasting context. Let Codex pull the right data from your systems of record and act through approved integrations
- 03 **Scale what works with skills**
Package reliable workflows as reusable “commands” for your team, so best practices show up on every project automatically
- 04 **Automate** recurring tasks in the background
Run on a schedule and post results to your Codex App inbox



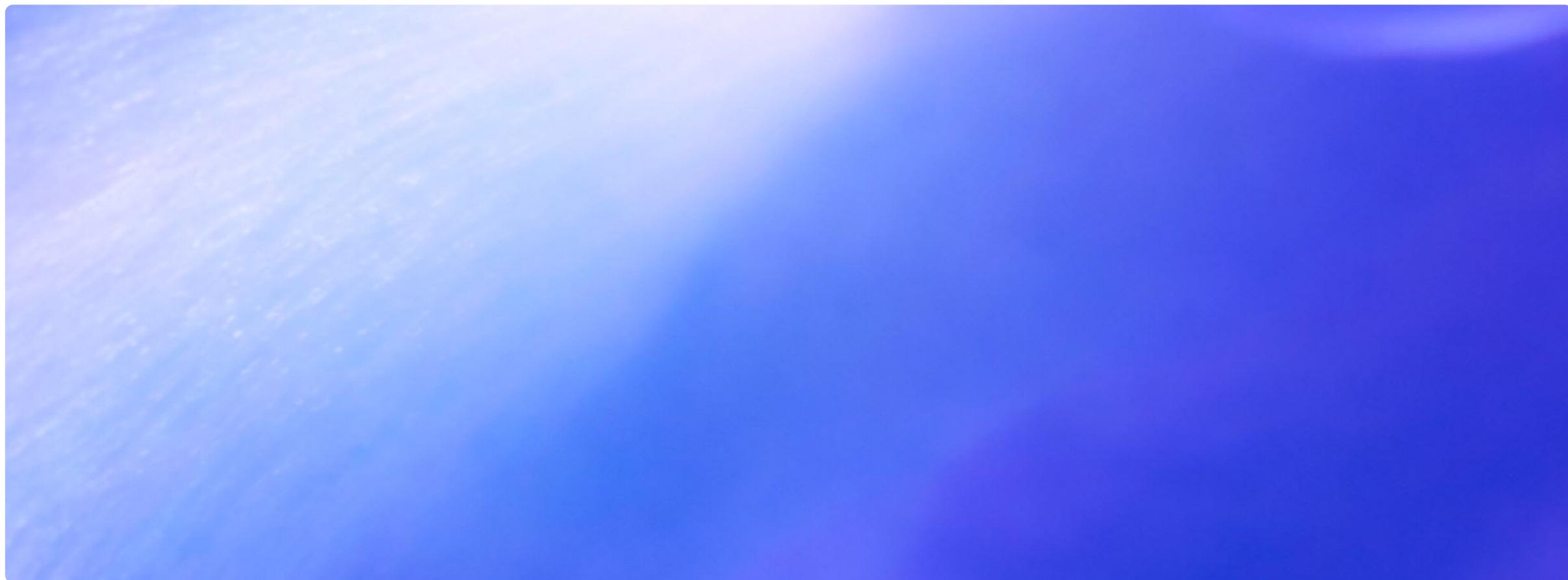
The Codex App is a Command Center for Parallel Workflows

Handle multi-agent workflows across many workspaces simultaneously

Use Skills to connect Codex to your tools and fit your teams workflows

Set up Automations to delegate recurring tasks to Codex in the background

Configurations



Configurations make Codex feel more like a teammate

- Codex reads local settings from the `~/.codex/config.toml` file
- Configs are shared between the CLI and IDE
- Can also be accessed in the IDE through Settings > Codex Settings > Open config.toml
- Set default model, reasoning, approval, sandbox, profiles, experimental features, MCPs etc or configure custom model providers

```
model = "gpt-5.2-codex"
model_reasoning_effort = "medium"
sandbox_mode = "workspace-write"
approval_policy = "on-request"

[features]
web_search_request = true
unified_exec = true //background terminal
shell_snapshot = true //avoid login scripts

[profiles.fast]
model = "gpt-5.1-codex-mini"
model_reasoning_effort = "medium"

[mcp_servers.openaiDeveloperDocs]
url = "https://developers.openai.com/mcp"

[shell_environment_policy]
include_only = ["PATH", "HOME"]
```

Secure by default: approval + sandboxing

approval_mode

(When does Codex ask for your permission to run a command?)

1. **untrusted:** Request permission when it attempts to use a command that's not on its "trusted" list
2. **on-failure:** If command fails, will ask to run outside sandbox.
3. (Default) **on-request:** Runs until it decides that it needs to ask for escalated permissions
4. **never:** Never ask for approvals

sandbox_mode

(Can Codex read / write in the directory, and which files can it access?)

1. **read-only:** Requires permission if attempt to write to file
2. (Default) **workspace-write:** Write to files in current directory
3. **danger-full-access:** Disable sandboxing altogether

- **Sandbox mode limits access:** By default, Codex can only edit the current workspace and cannot use the network
- **Approval mode gates actions:** Codex pauses when it needs elevated permissions, so you stay in control
- **Escalate intentionally:** Loosen these settings only for trusted repos or specific tasks

Reference: <https://developers.openai.com/codex/security>

Multi-agents

- Enable multi-agents in the Codex settings
~/`.codex/config.toml` file or `/experimental`
- Each agent has a description, distinct developer instruction, and model + reasoning settings, and sandboxing
 - Built-in roles: `default`, `worker`, `explorer`, `monitor`
- E.g. `reviewer`

```
model = "gpt-5.3-codex"  
model_reasoning_effort = "high"  
developer_instructions = "Focus on high priority  
issues, write tests to validate hypothesis before  
flagging an issue. When finding security issues give  
concrete steps on how to reproduce the vulnerability."
```



```
[features]  
multi_agent = true
```



```
[agents.default]  
description = "General-purpose helper."  
  
[agents.reviewer]  
description = "Find security, correctness, and test risks  
in code."  
config_file = "agents/reviewer.toml"  
  
[agents.explorer]  
description = "Fast codebase explorer for read-heavy  
tasks."  
config_file = "agents/custom-explorer.toml"
```

Multi-agents

- ✓ Define investigation slices and assign them to multiple explorer agents
- Launch awaiter agents to monitor any long-running repo inspection commands
- Synthesize agent findings into a codebase map and key follow-up areas

The agent fan-out is running now. I'm waiting on all six in parallel so we can synthesize once the first wave of findings is complete instead of polling them one by one.

• Waiting for 6 agents

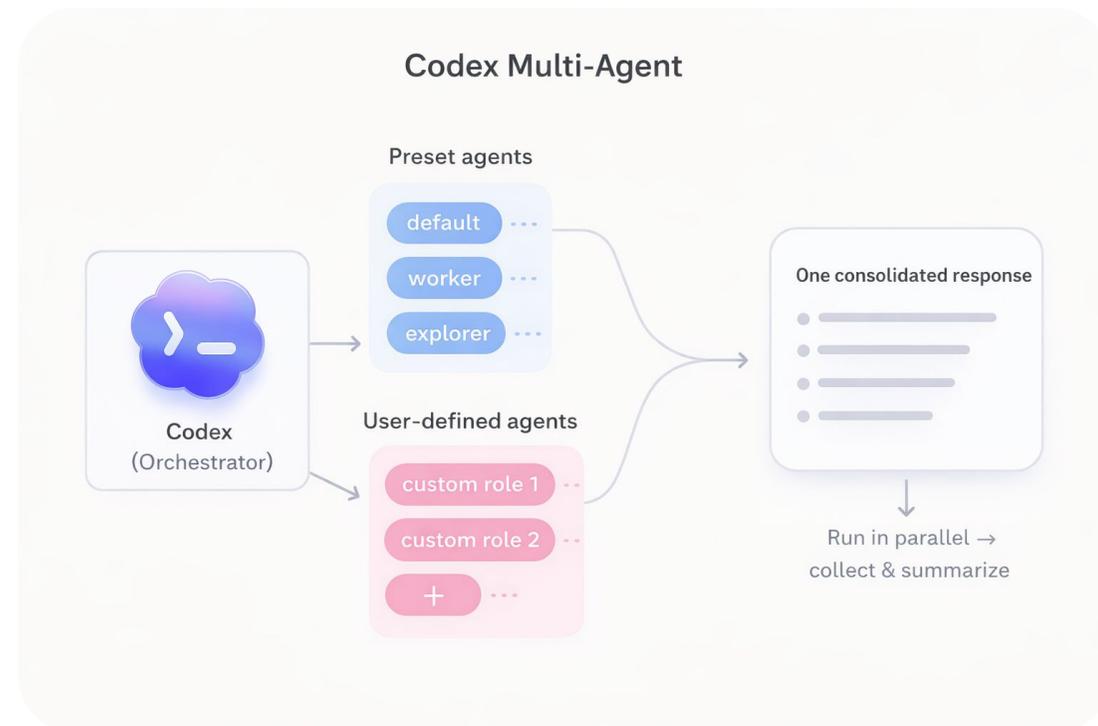
- └ Sage [explorer]
- └ Spruce [explorer]
- └ Yew [explorer]
- └ Camellia [explorer]
- └ Ash [awaiter]
- └ Date [awaiter]

Agents

Select a thread to focus

- > 1. • Main [default] (current) 019c954b-fdda-7d92-a103-3e21176cc5c1
- 2. • Sage [explorer] 019c9551-d2c0-7892-b4af-8759f4edc30f
- 3. • Spruce [explorer] 019c9551-d3b2-7e11-aaf8-1d93cf7ad6ce
- 4. • Yew [explorer] 019c9551-d470-74f1-bca9-1b6a9dbc5bbd
- 5. • Camellia [explorer] 019c9551-d510-72f3-bfa7-db33d4003d38
- 6. • Ash [awaiter] 019c9551-d5e3-73a0-9b8e-ddd1f54a2d9f
- 7. • Date [awaiter] 019c9551-d699-7d53-8455-fcc00179f37e

Press enter to confirm or esc to go back



Memory

- Enable memory in the Codex settings
`~/.codex/config.toml`
- Idle sessions in the background are consolidated into a local memory workspace.
- Memories then influences future sessions by injecting a compact summary and making deeper memory files available for lookup.
- Codex can better carry forward durable preferences, recurring project context, and prior decisions across sessions.



```
[features]
memories = true
```



```
~/.codex/memories/memory_summary.md

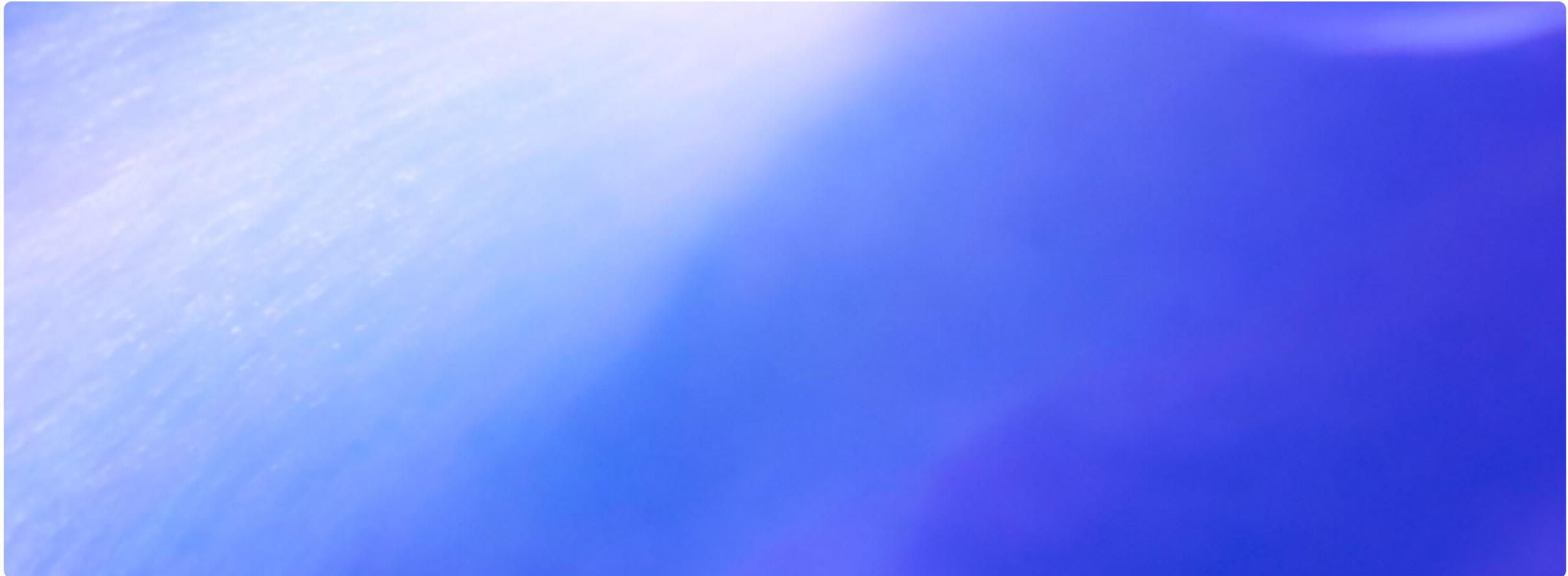
## User Profile
This user uses Codex as an
execution-heavy engineering workbench,
not a brainstorming toy...
```

Advanced configuration reference

Key	Type / Values	Details
<code>approval_policy</code>	<code>untrusted</code> <code>on-failure</code> <code>on-request</code> <code>never</code>	Controls when Codex pauses for approval before executing commands.
<code>chatgpt_base_url</code>	<code>string</code>	Override the base URL used during the ChatGPT login flow.
<code>check_for_update_on_startup</code>	<code>boolean</code>	Check for Codex updates on startup (set to false only when updates are centrally managed).
<code>cli_auth_credentials_store</code>	<code>file</code> <code>keyring</code> <code>auto</code>	Control where the CLI stores cached credentials (file-based <code>auth.json</code> vs OS keychain).
<code>compact_prompt</code>	<code>string</code>	Inline override for the history compaction prompt.
<code>developer_instructions</code>	<code>string</code>	Additional developer instructions injected into the session (optional).
<code>disable_paste_burst</code>	<code>boolean</code>	Disable burst-paste detection in the TUI.
<code>experimental_compact_prompt_file</code>	<code>string (path)</code>	Load the compaction prompt override from a file (experimental).
<code>experimental_instructions_file</code>	<code>string (path)</code>	Experimental replacement for built-in instructions instead of <code>'AGENTS.md'</code> .

Reference: <https://developers.openai.com/codex/config-advanced>

MCP

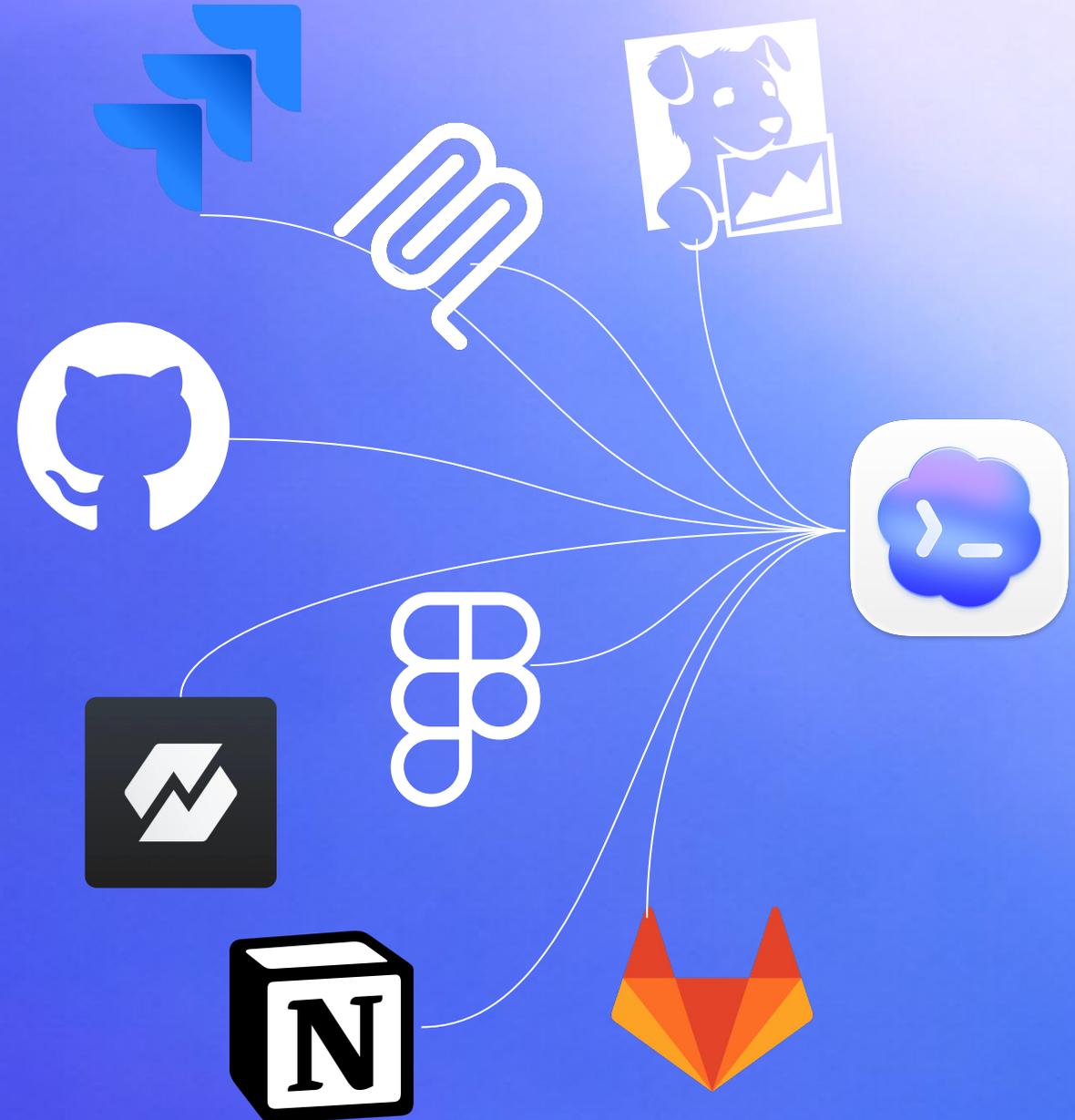


Model Context Protocol (MCP)

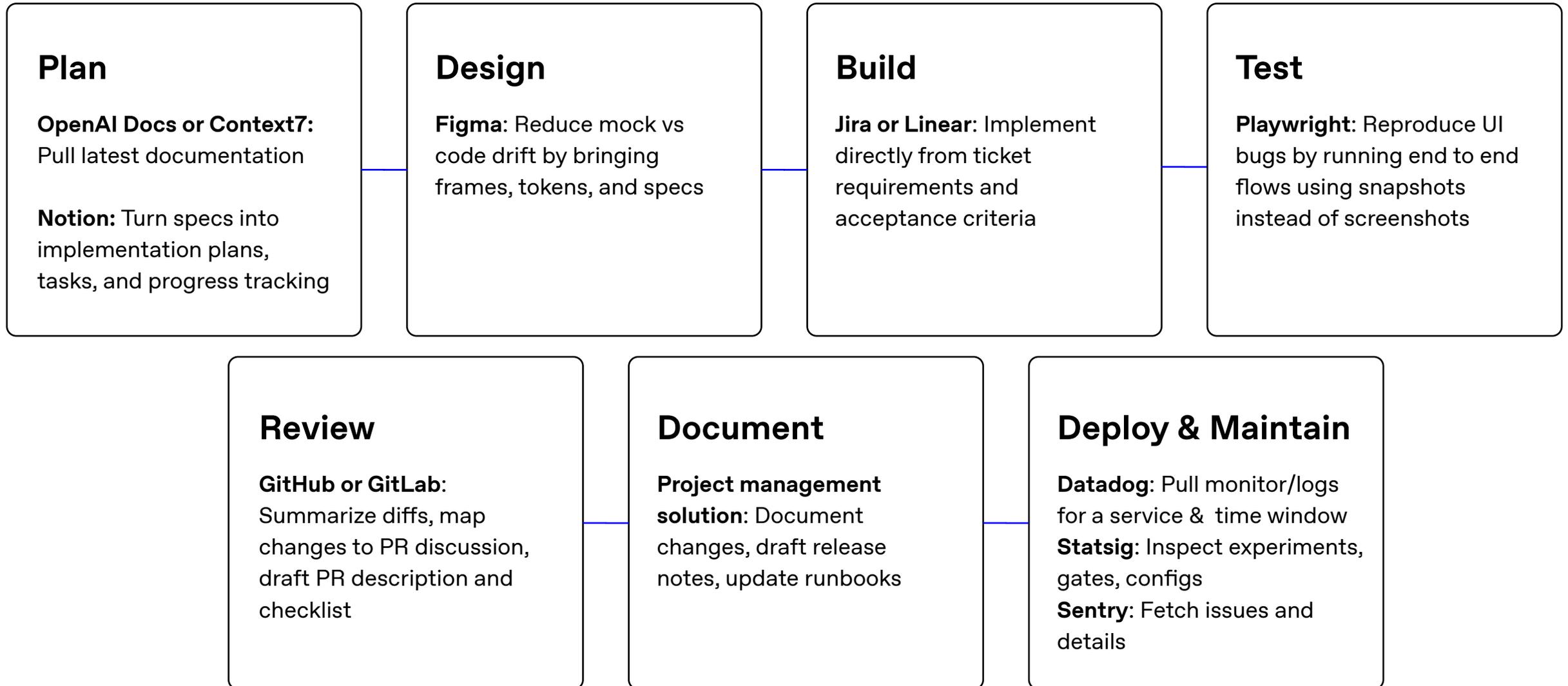
- Open standard designed for portable tool and context integrations
- MCPs can be hosted internally or remotely by other companies
 - STDIO: Servers that run as a local process
 - Streamable HTTP: Servers accessed at an address
- Add with `codex mcp add` command

Pro: Tools portable between different agent harnesses

Con: Less control over context pollution by the MCP



MCPs examples across the software lifecycle



Example: add OpenAI docs MCP

Let Codex access the OpenAI Documentation

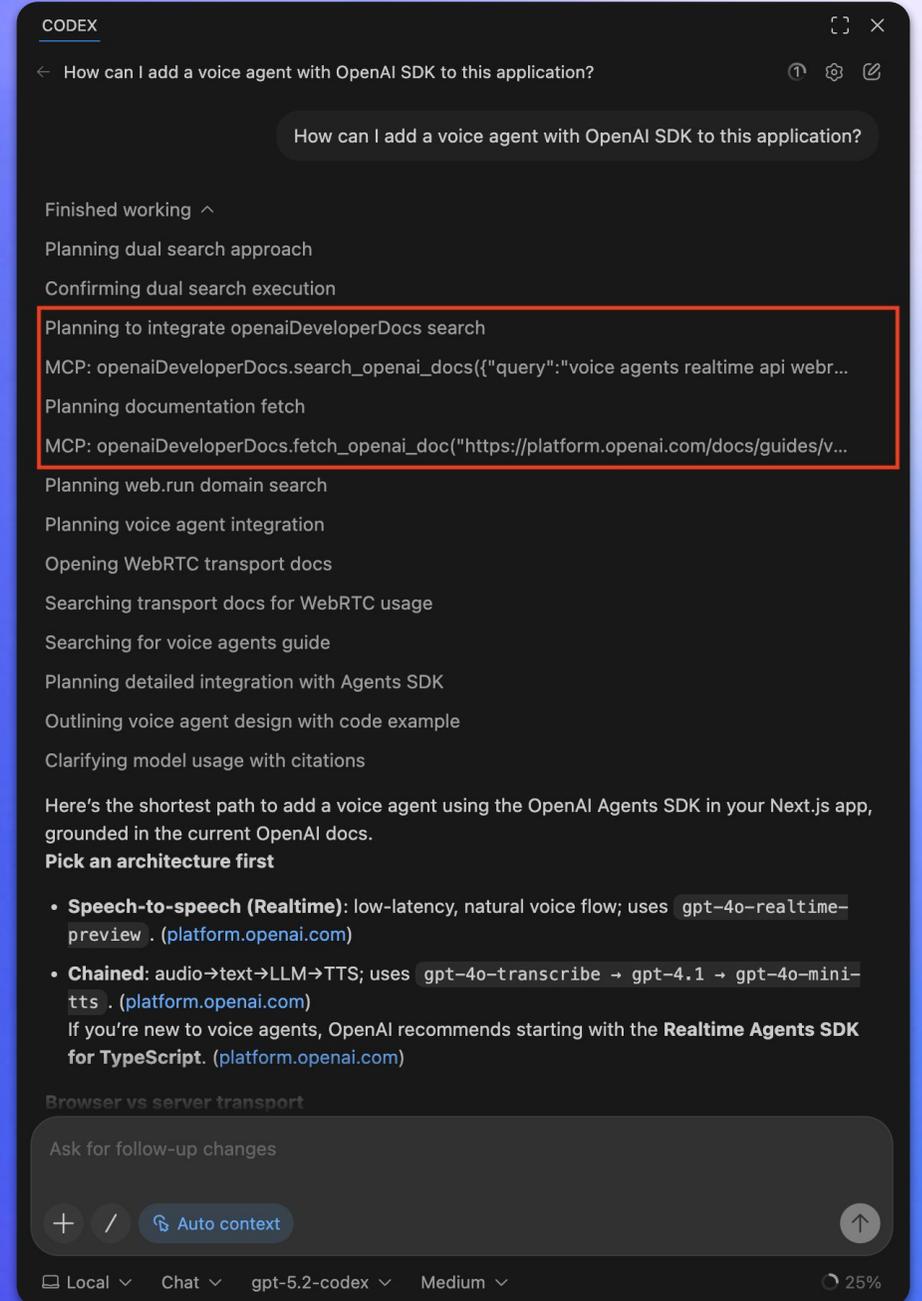
```
codex mcp add openaiDeveloperDocs --url https://developers.openai.com/mcp
```

Prompt

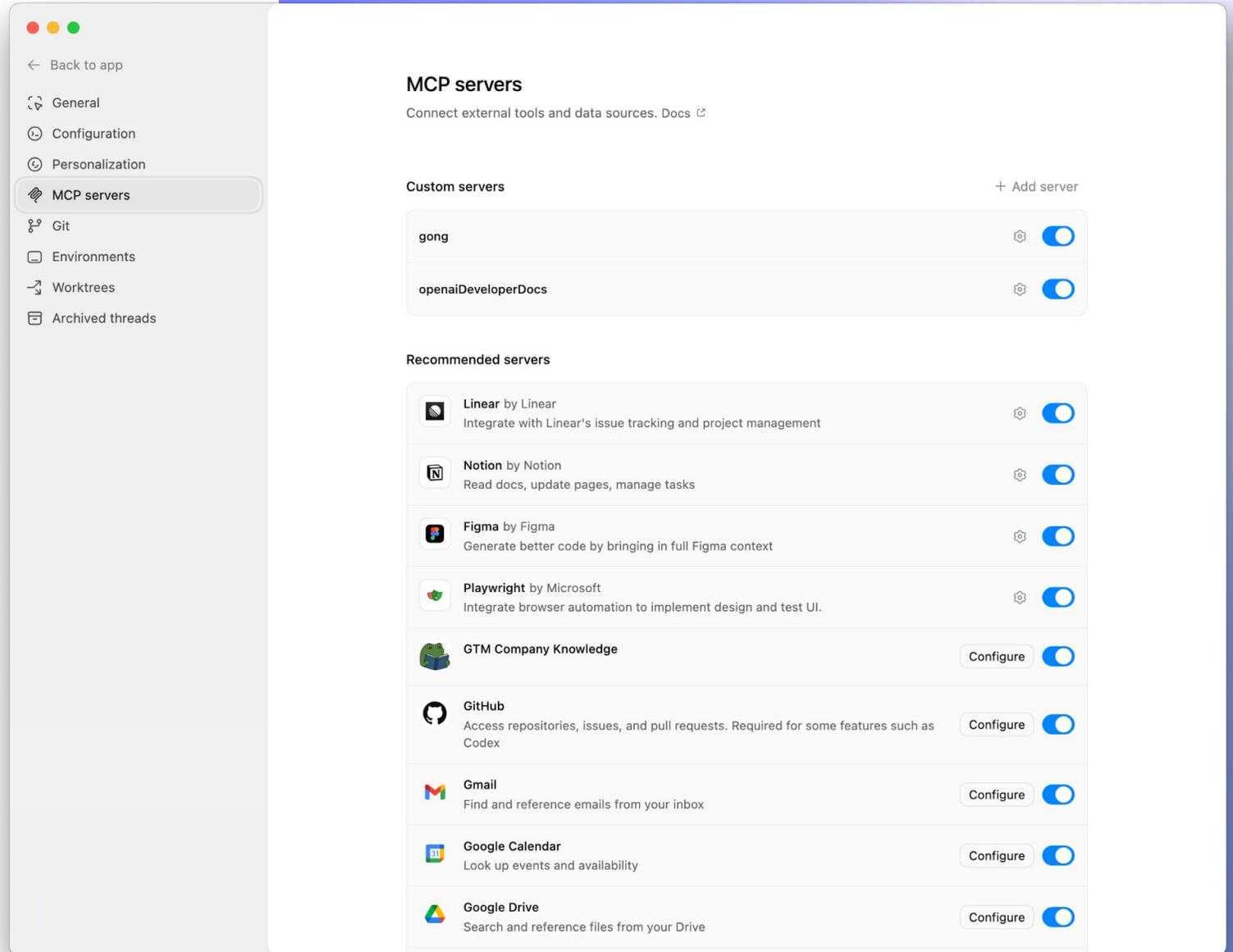
How can I add a voice agent with OpenAI SDK to this application?

Optional - Add this snippet to your AGENTS.md

```
Always use the OpenAI developer documentation MCP server if you need to work with the OpenAI API, ChatGPT Apps SDK, Codex, or related docs without me having to explicitly ask.
```



Connect data sources for your skills to use in Settings → MCP Servers



Build Your Skills



Introduction to skills

- Use skills to share team workflows and best practices in Codex
- Difference: Skills bundle full workflows (instructions, resources, scripts) vs MCP only connects to external systems
- Core file: **SKILL.md** with metadata (name, description) plus instructions
- Optional folders: `scripts/`, `references/`, `assets/` for code, docs, and templates

```
▼ my-skill/  
├── SKILL.md           Required: instructions + metadata  
├── scripts/          Optional: executable code  
├── references/       Optional: documentation  
└── assets/           Optional: templates, resources
```

How to use skills

Automatic Usage:

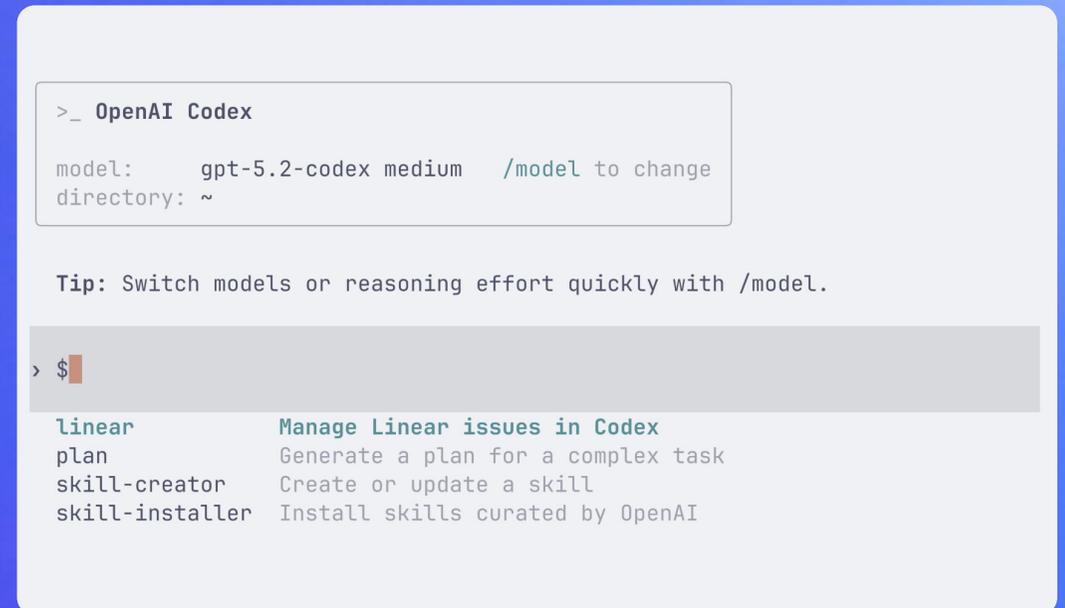
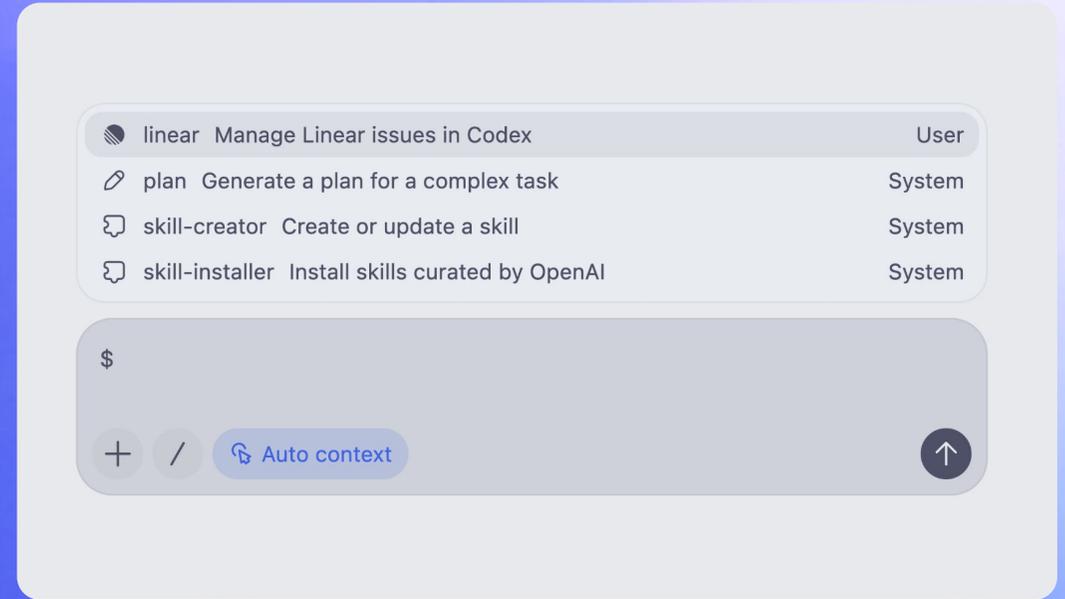
- Codex can see skill descriptions
- Automatically chooses relevant ones.

Invoke skills:

- Explicit: `$skill-name` or `/skills`

Install skills:

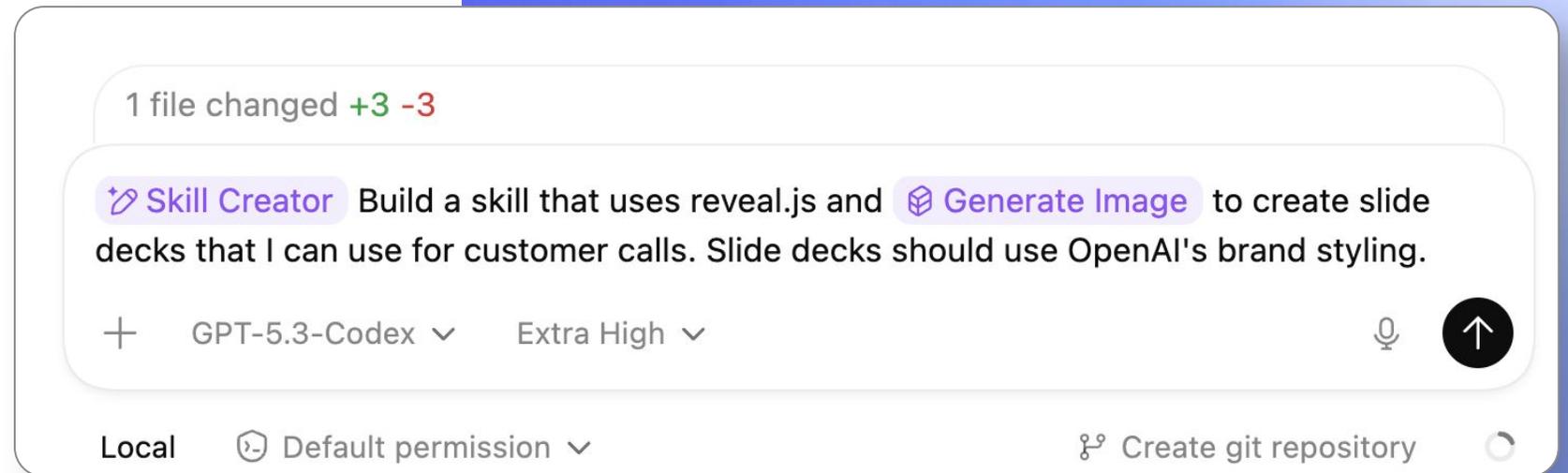
- Use `$skill-installer` to install curated skills or from a repository URL



How to create skills

Use the **\$skill-creator** skill to create the first draft

Skills can even invoke other skills



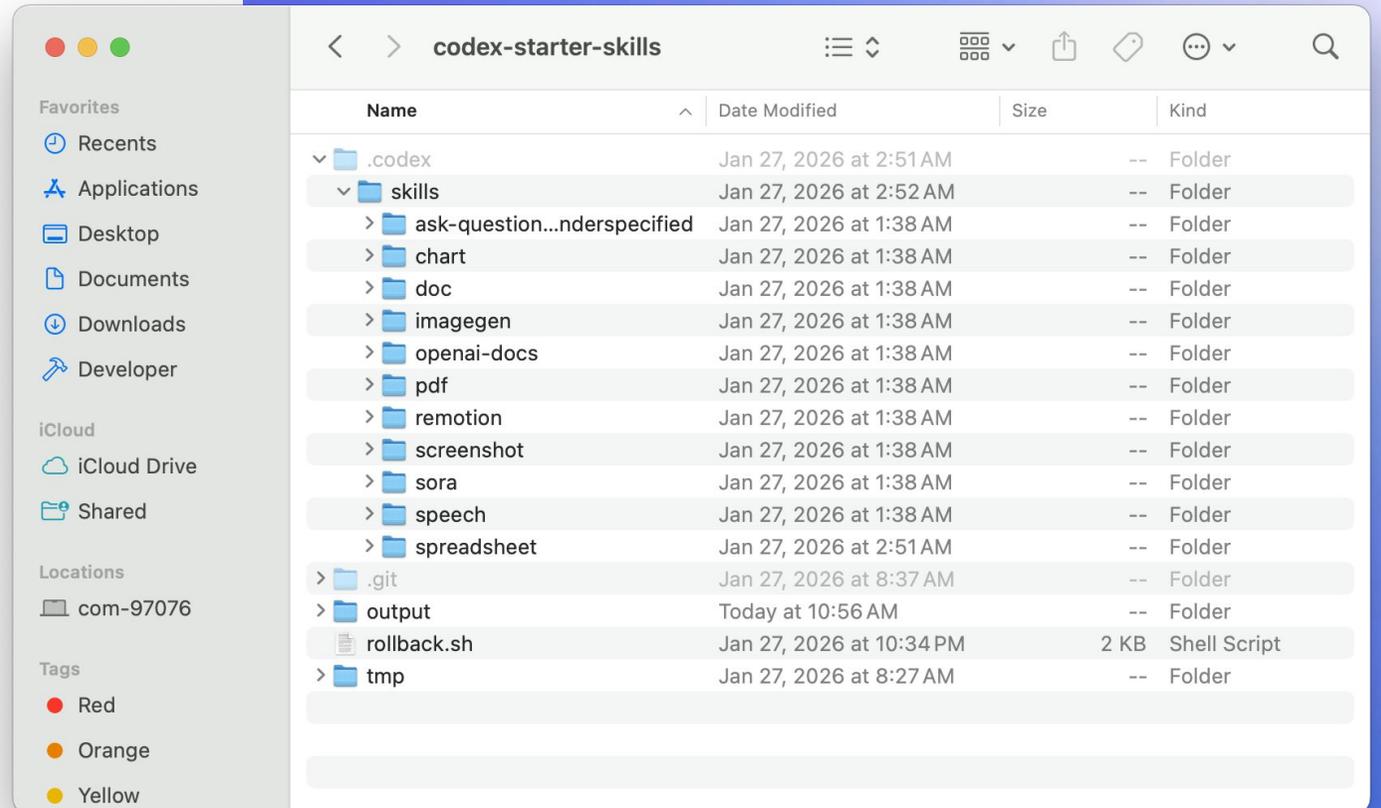
Where skills are saved

Codex reads skills from repository, user, admin, and system locations

Common locations

- `~/.codex/skills/<skill-name>` (user)
- `./codex/skills/<skill-name>` (repo)

More options [here](#)



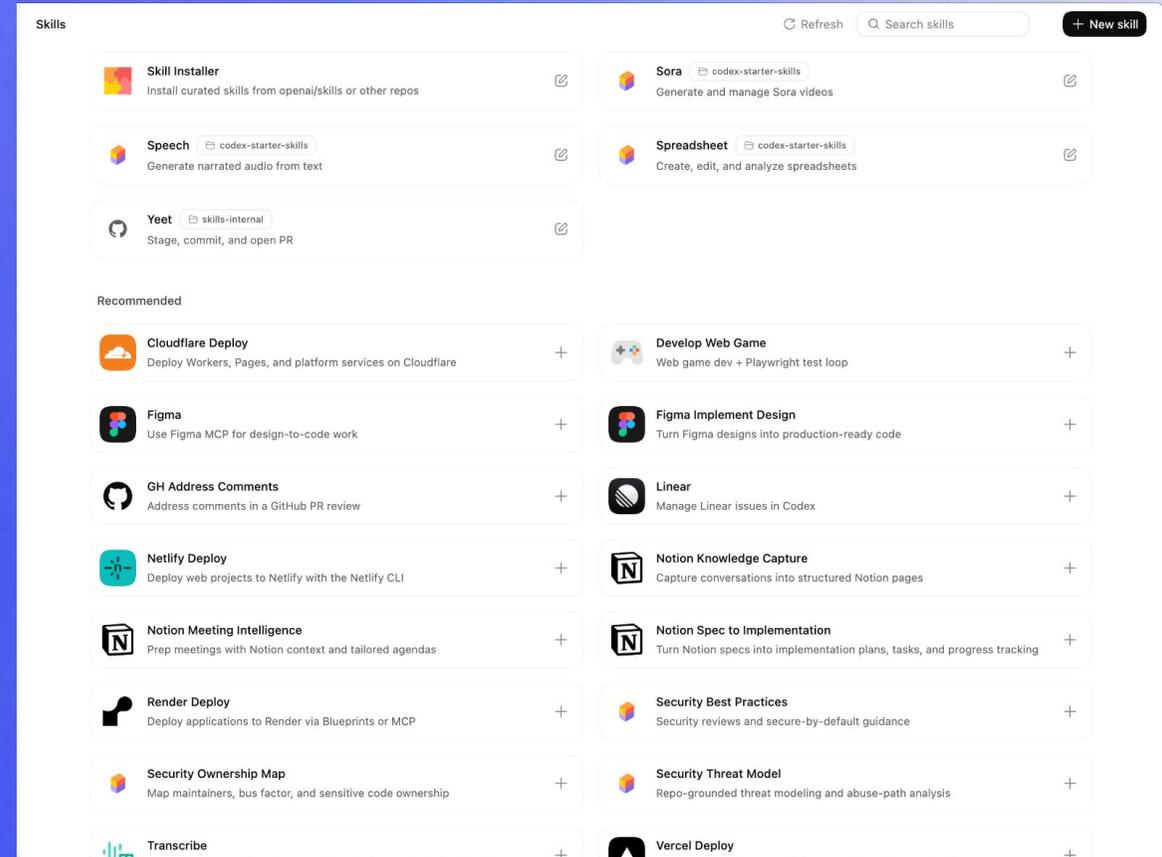
Build a Skill for your team

Goal: Everyone leaves with **one real Skill** that makes *your* team faster (not a toy demo).

How to pick the right Skill

- Pick a task **agents struggle with** today, but your team does constantly
- Or pick a task that requires **access/structure agents don't naturally have:**
 - internal tools
 - CLIs
 - “how we do it here” guardrails (compliance, tagging, workflows)

Examples: <https://github.com/openai/skills>



Ask Codex for help

What “done” looks like

- It encodes your team’s **best practice + workflow + validation** .

If you’re stuck, build this default Skill

- **Jira Daily Triage Skill:** reads today’s Jira tickets, summarizes, proposes next actions, and drafts updates.

Other strong examples

- **AWS Infra Skill:** enforces tagging, security rules, and “safe-by-default” setup.
- **Repo Quality Skill:** testing conventions, CI checks, monitoring/release steps.

```
---  
name: skill-name  
description: Description that helps Codex select the skill  
metadata:  
  short-description: Optional user-facing description  
---
```

Skill instructions for the Codex agent to follow when using this skill.

Skills can be used across technical / non-technical teams

Autofix CI

Given a Buildkite build URL, download failing job logs, read the error output, map the issue to the local codebase, and propose a concrete fix with testing guidance.

GitHub Address Comments

Help address issue comments on the open GitHub PR for the current branch using gh CLI

Gong Transcript Reader

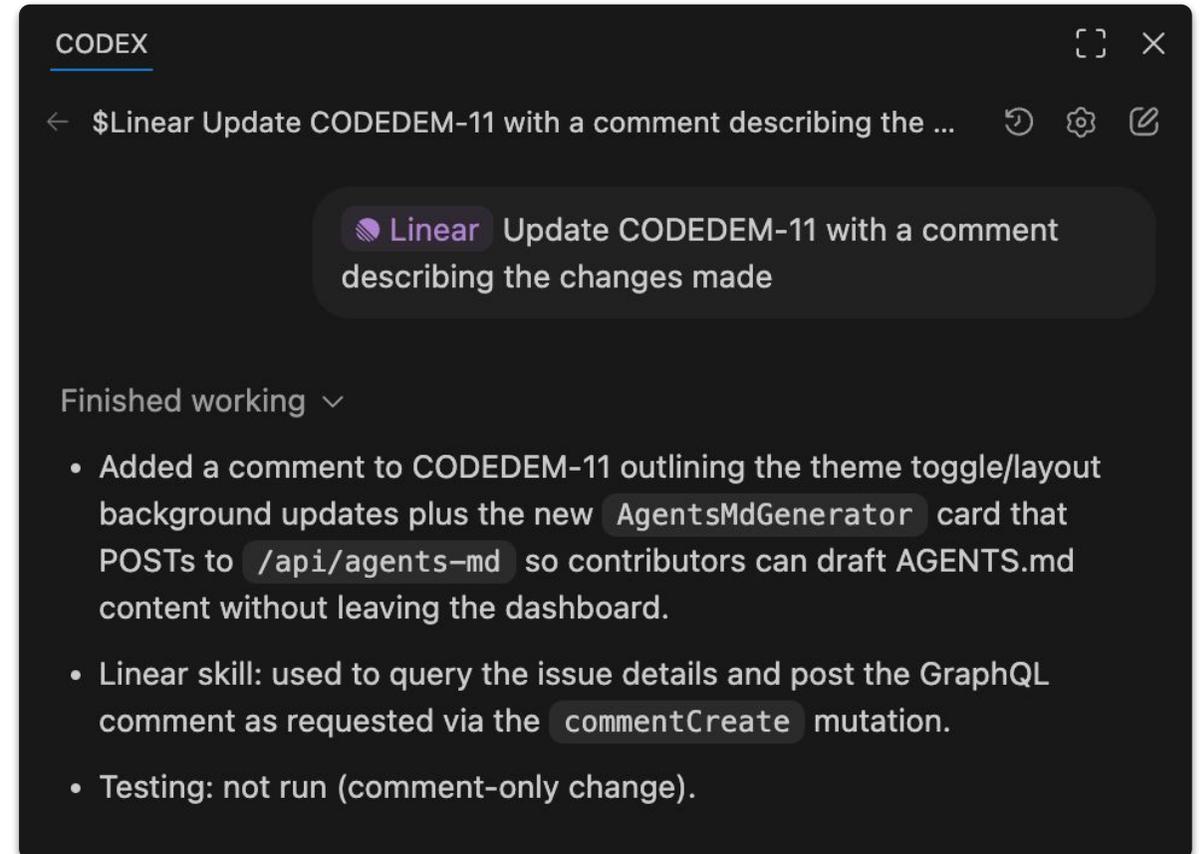
Get Gong transcripts from recent calls and Codex analyzes GTM member's potential weak points based on conversations

Spreadsheets

Create, edit, analyze and format spreadsheets using Python with option rendering for visual checks.

Keep documentation in sync with code

- Generate first pass docs from diffs: module summaries, API notes, and “what changed” writeups
- Draft release notes automatically: from commits and PRs so updates ship with the code
- Post updates where work lives: comment on the ticket/PR with a crisp summary and next steps



Codex summarizes the change and posts it back to Linear for team visibility

From issue to draft PR (end to end)

A reusable skill that standardizes how teams ship changes

Let's use the **\$gh-issue-plan-pr** skill to:

- 1) Triage the GitHub issue
- 2) Create a plan with **\$create-plan**
- 3) Execute edits
- 4) Document the changes
- 5) Launch draft PR with **\$launch-draft-pr**

Workflow: Plan → Build → Test
→ Review → Document → Deploy

The image shows a workflow for creating a draft pull request from a GitHub issue. It consists of two main parts: a terminal window and a GitHub pull request interface.

Terminal Window (Top):

- Command: `gh-issue-plan-pr` Pick up issue #136: N+1 query when fetching tickets for all users from oaiagicorp/agent-desk
- Status: Finished working
- Message: Using gh-issue-plan-pr. I pulled issue #136 and drafted a PR. Proceed with the edits?

Terminal Window (Bottom):

- Status: Finished working
- Message: Updated the ticket list API to include comment counts, removed per-ticket detail fetch in the table, and added a regression test to ensure the UI no longer triggers N+1 fetches.
- Changes:
 - `route.ts` now includes `_count.comments` alongside `assignee/author`.
 - `route.ts` matches the list shape with `_count.comments`.
 - `ticket-table.tsx` uses comment counts and drops the per-ticket fetch.
 - `ticket-table.test.tsx` asserts no per-ticket fetches, even with 12 tickets.

GitHub Pull Request (Right):

- Title: Fix N+1 ticket list queries #137
- Status: Draft
- Author: derrickchoi-openai
- Target: main
- Source: fix/n1-ticket-list-136
- Files changed: 4
- Summary:
 - include comment counts in ticket list/search responses for eager loading
 - remove per-ticket detail fetches in the ticket table
 - add regression test to ensure large lists do not fetch per ticket
- Tests:
 - npm test -- tests/ticket-table.test.tsx
- Tickets:
 - N+1 query when fetching tickets for all users #136

Automations



Automate your workflows

What is it?

Ability to codify workflows and tasks. Runs on your computer

Why does it matter?

Teach Codex how to do a task or workflow right the first time

Automations Beta

Automate work by setting up scheduled threads. [Learn more](#)

Start with a template



Scan recent commits (since the last run, or last 24h) for likely bugs and propose minimal fixes.



Draft weekly release notes from merged PRs (include links when available).



Summarize yesterday's git activity for standup.



Summarize CI failures and flaky tests from the last CI window; suggest top fixes.



Create a small classic game with minimal scope.



From recent PRs and reviews, suggest next skills to deepen.



Synthesize this week's PRs, rollouts, incidents, and reviews into a weekly update.



Compare recent changes to benchmarks or traces and flag regressions early.



Detect dependency and SDK drift and propose a minimal alignment plan.



Identify untested paths from recent changes; add focused tests and use \$yeet for draft PRs.



Before tagging, verify changelog, migrations, feature flags, and tests.



Update AGENTS.md with newly discovered workflows and commands.



Summarize last week's PRs by teammate and theme; highlight risks.



Triage new issues; suggest owner, priority, and labels.



Check CI failures; group by likely root cause and suggest minimal fixes.



Scan outdated dependencies; propose safe upgrades with minimal changes.



Audit performance regressions and propose highest-leverage fixes.



Update the changelog with this week's highlights and key PR links.

Improve skills using automations

- Add a regular task to read past sessions in `~/.codex/sessions` and use them to update/create new skills for **repetitive/common tasks**
- Result: Continuously improving Skills or AGENTS.md
- You can always manually ask Codex to do a retrospective and update the Agents.md for you to avoid future mistakes or get to an end result faster

Create automation

Automate recurring tasks in the background. Codex adds findings to the inbox, or automatically archives the task if there's nothing to report. [Learn more](#)

Name

Update AGENTS.md

Projects

If you want an automation to run on a specific branch, you can specify it in your prompt.

codex-vscode ×

Automations run in the background on dedicated worktrees.

Prompt

Update AGENTS.md with newly discovered workflows and commands.

Constraints:

- Keep edits minimal, accurate, and grounded in repo usage.
- Do not touch unrelated sections or auto-generated files.
- If you are unsure, prefer adding a TODO with a short note rather than inventing.

Schedule

Daily Interval

09:00 AM



Mo

Tu

We

Th

Fr

Sa

Su

Automations run with your default sandbox settings. Tool calls will fail if they require modifying files outside the workspace, accessing network, or working with apps on your computer. You can selectively allowlist commands to run outside the sandbox using [rules](#).

Cancel

Create

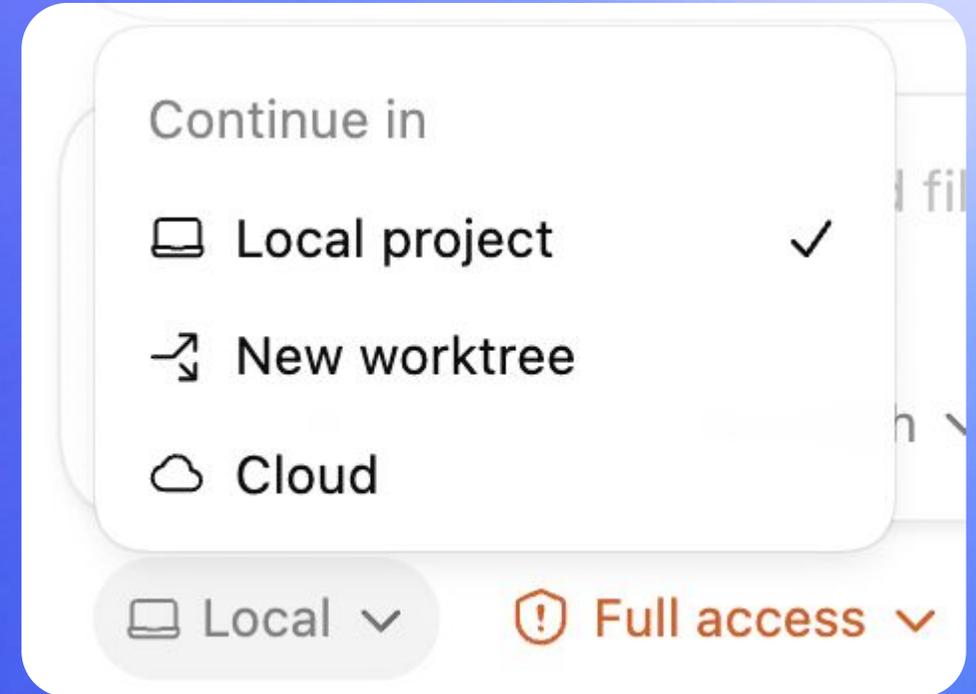
Bonus: Worktrees



Worktrees: Parallel copies of your codebase

- Worktrees keep changes isolated so multiple agents can work on the codebase without overwriting each other
- Review like you already do: diff view, inline comments, stage or revert chunks, then commit, push, and open a PR directly in app

Why it matters: You can run many agents at once without edit conflicts



Recap: A **New** Operating Model for Engineering

01 **Set Agents up for Success**

Safely transition engineering organizations from writing code to delegating tasks to agents across the SDLC

02 **Establish Standards & Control**

Set guardrails and permissions for agents, decide where and how Codex can operate, and maintain visibility into actions & outputs

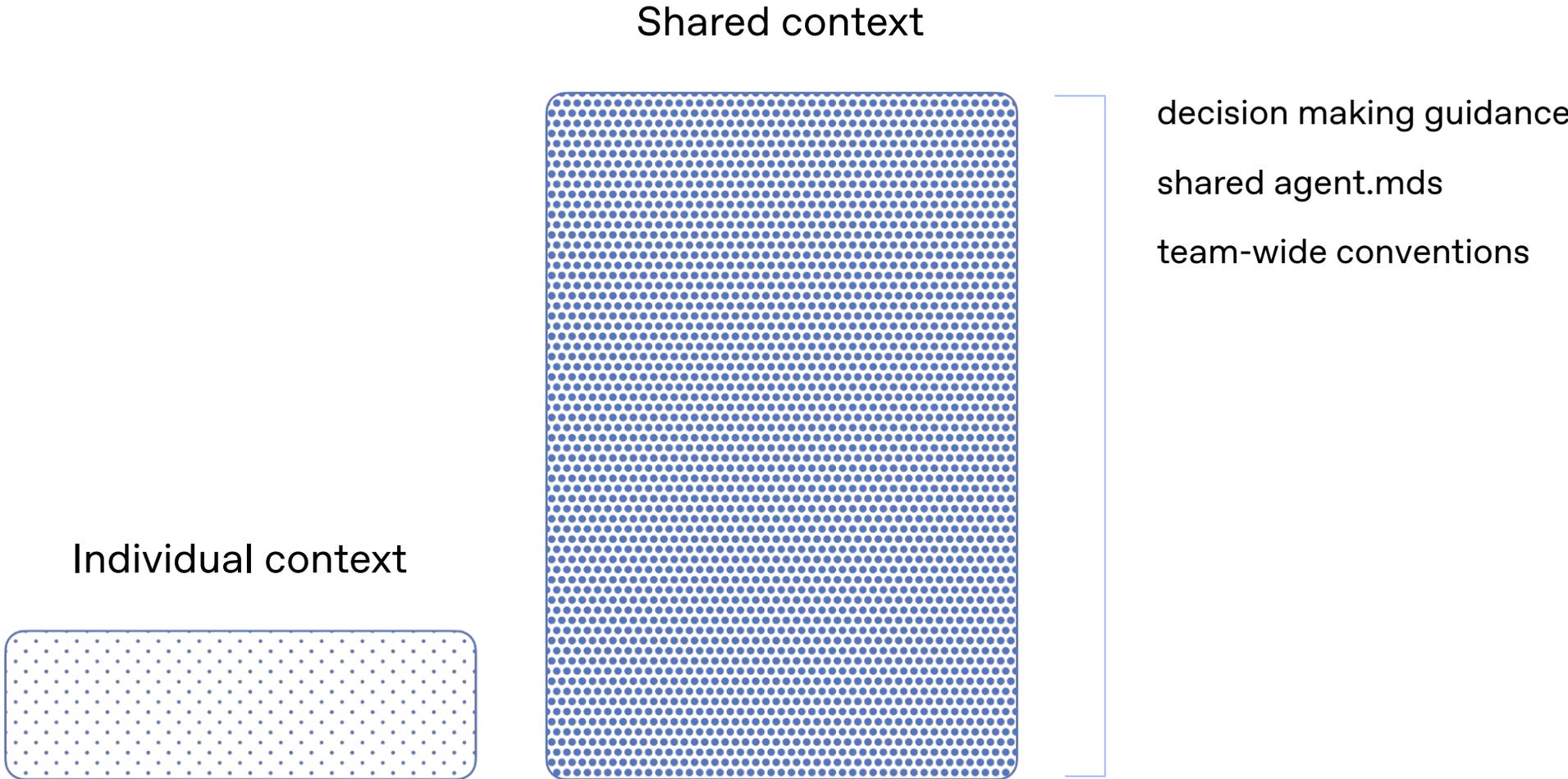
03 **Connect with your existing tools**

Integrate with Slack, Datadog, Jira, GitHub to pull the right context automatically & reduce friction across systems

03 **Turn best practices into repeatable workflows**

Package reliable workflows as reusable “commands” for your team, distribute them across your org, and ensure consistent quality/standards

Shared context helps you scale



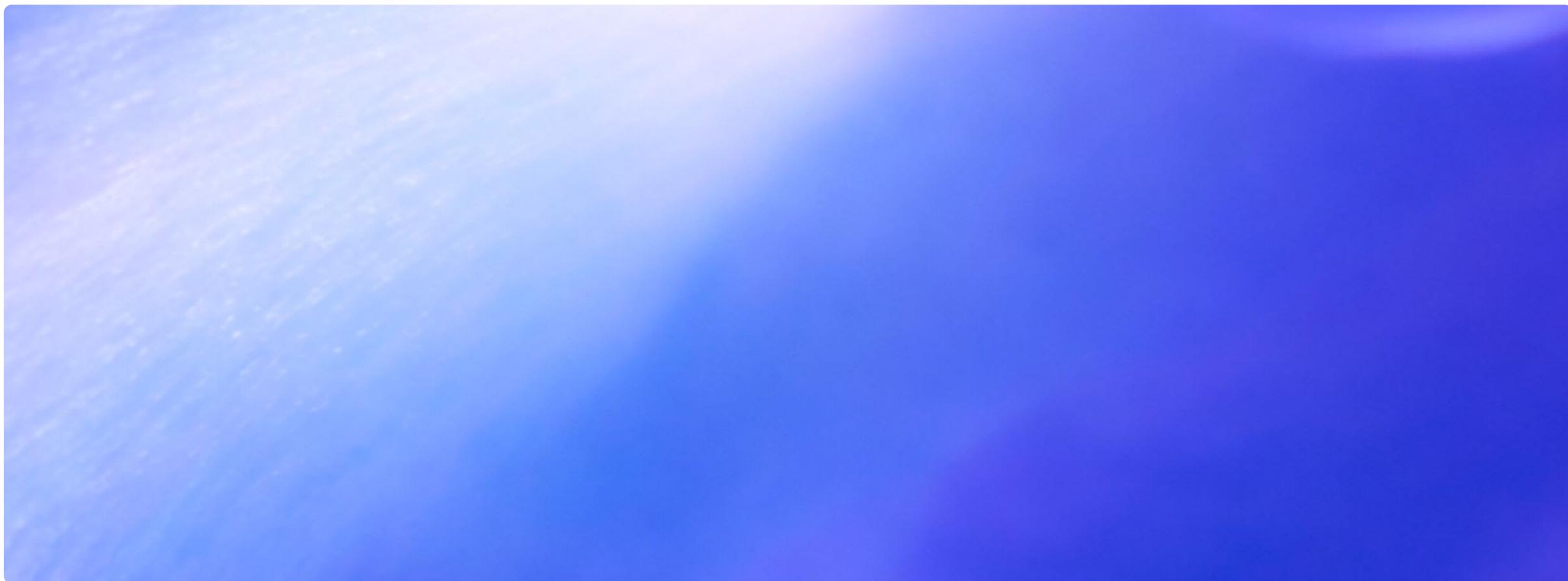
Next Steps

- Set up your configurations
- Connect to your tools and MCPs
- Create skills and automations for you and your team



Q&A

OpenAI



Resources

Codex Documentation

developers.openai.com/codex



Codex Cookbooks

cookbook.openai.com/topic/codex



Codex Changelog

developers.openai.com/codex/changelog

