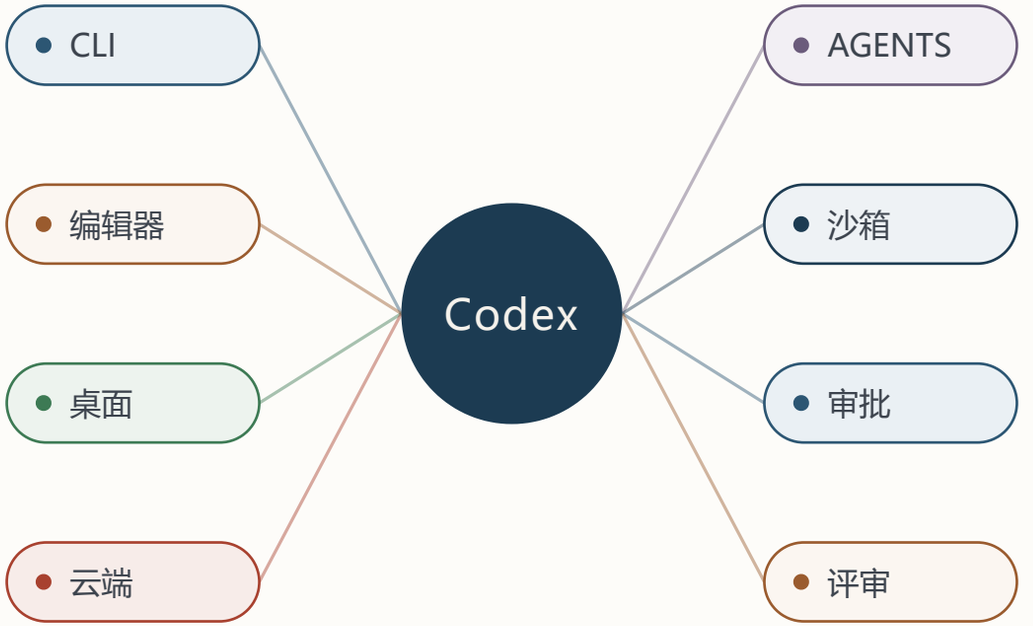
COURSE 06 · CODEX

CODEX 企业实操课

从装上一个入口，到团队有一份自己的规矩



AI 教学课件工厂 · 系列课程 06

企业研发与测试团队 / 约 90 分钟

它给的是 DIFF，不是一行补全

补全给你一行，它给你一个 diff。

01 你给一句话，它自己找文件、改代码、跑测试，最后把改动摊给你看

02 它读的是你真实的仓库，不是你贴进去的片段

03 四个入口共用同一套配置和会话：终端、编辑器、桌面、云端

04 它的上限取决于拿到的 shell 和权限，不是模型参数



先把它当成一个新同事，而不是一个更聪明的 Tab 键。

四个入口，别装重了

同一个智能体的四道门，装一个就够开始。

01 终端 CLI：每一步都看得见，适合日常改代码和排查

02 编辑器扩展：改动直接以 diff 出现在你正在看的文件上

03 桌面应用：多个任务并行，各自一个独立工作区

04 云端：扔上去就走，跑完给你一个分支或 PR

CLI
终端里，可监督
日常改动

编辑器
改动即 diff
小改动

桌面
多任务并行
跨模块重构

云端
异步，不占本机
长任务

想盯着看用 CLI，想撒手用云端。

一次任务，四个动作

它做的事和你自己做的一样，只是不用你敲。

01 先读：从 AGENTS.md 开始，再按需打开相关文件

02 再改：在工作区里直接写文件，不是给你一段代码让你贴

03 再验：跑你项目里已有的构建和测试命令

04 最后交：diff、命令日志、测试输出一起给你



它交的是「改完了，测试过了」，不是「你可以试试这样」。

同一个模型，在别人仓库里更听话

差别通常就在一份文件上。

01 AGENTS.md 是它每次动手前必读的项目说明

02 里面写构建命令、测试命令、代码约定、不许碰的地方

03 不写，它就按通用习惯猜，猜错一次你要改半天

04 这份文件进版本库，团队共用，不是个人配置

没有 AGENTS.md	写好 AGENTS.md
按通用习惯猜	构建测试命令明确
测试命令猜错	约定写死
改了不该改的	禁区标出来

它猜错的成本，就是你写 AGENTS.md 的理由。

四段就够，别写成文档

写长了每次都要读，费的是你的额度和时间。

01 怎么装、怎么跑、怎么测：三条命令

02 代码约定：命名、目录结构、错误处理、日志

03 禁区：别动生成的代码、别改迁移文件、别碰密钥

04 评审规则：PR 里什么算 P0、什么算 P1



只写「不看就会做错」的东西，其余删掉。

它按目录往下找，越近的越优先

根目录一份，子目录可以各写一份。

01 先读全局的 ~/.codex/AGENTS.md，再按目录从根往下读

02 同一个目录里，override 文件优先于普通文件

03 越靠近当前目录的文件越优先，冲突时覆盖前面的

04 全部加起来默认上限 32 KiB，超了后面的读不到

全局
~/.codex/AGENTS.md
所有项目

根目录
仓库根的一份
团队共用

子目录
就近的覆盖远的
模块约定

上限
默认 32 KiB
超了读不到

上限可以调，但先想想为什么写到 32 KiB 还不够。

别从空白写，让它先起草

/init 会扫一遍仓库，生成一份初稿。

01 在项目目录里跑 /init，它按现有代码猜构建和测试方式

02 初稿一定要人工改：猜错的地方比漏掉的多

03 改完提交进版本库，以后每次任务都受益

04 用一句话让它自查：「总结当前加载的指令」，确认读到了



生成的初稿是草稿，不是配置。

三种权限，从最紧的开始

它能做什么由配置决定，不由你叮嘱决定。

01 read-only：只能读，适合让它先分析、出方案

02 workspace-write：能改工作区，是版本库里的默认档

03 danger-full-access：什么都能碰，只在隔离环境里用

04 升档要有理由，不是嫌麻烦就往上调

read-only
只读，不改文件
先分析

workspace-write
只改工作区
日常默认

full-access
不设限制
隔离环境

默认给 workspace-write，别给 full-access。

什么时候让它停下来问你

审批管的是动手前要不要报备，和沙箱是两件事。

01 按需询问：要跑命令、要联网时停下来等你，是默认档

02 全程不问：适合已经跑通的重复任务，效率最高

03 逐条确认：改一次问一次，适合动核心模块

04 网络访问默认是关的，要用得显式打开



第一次跑的任务别关审批，跑顺了再关。

三条线，写进文件也写进制度

配置能拦住一部分，制度要拦住剩下的。

01 密钥和环境变量不进仓库，也不进提示词

02 生产库、生产账号不给它碰，只给本地或测试环境

03 客户数据不贴进云端任务，要用就走内网或本地

04 每次任务前先提交，出问题才能一键回退

密钥 不进仓库与提示词	零容忍
生产环境 只给测试与本地	隔离
客户数据 不进云端任务	内网或本地
提交点 动手前先 commit	可回退

它能回退的前提，是你先提交了。

一次一件事，做完就提交

一次让它改十个文件，出错你都不知道从哪查。

01 一个任务一个目标：修这个 bug、加这个接口

02 先 commit 再让它动手，diff 才是干净的

03 大改造拆成几个任务，一个一个过

04 它改完先看 diff，再跑测试，别直接合



任务的边界，就是你能验收的边界。

非交互模式：给流水线用

同一套能力，去掉对话框，接进脚本。

01 codex exec "任务描述" 跑完就退出，不等人

02 加 --json 输出结构化结果，便于流水线判断

03 适合：批量升级依赖、写迁移脚本、按报错起草修复

04 CI 里跑是自动的，预算要先设上限



能进 CI 的前提是任务可判定，不是「看着还行」。

让它先看一遍 PR

它擅长挑无聊的错，把人的时间留给设计。

01 在 PR 里 @codex review 触发一次评审

02 只标 P0 和 P1，不会给你一堆风格意见

03 评审规则写在 AGENTS.md 的评审段里

04 让它顺手修：@codex fix the P1 issue



自动评审是省人力，不是省责任，合并还是人点。

额度是区间，不是固定次数

同样一条消息，读三个文件和读三百个文件不是一个量级。

01 按席位订阅，团队版按人算；程序调用走 API 按量

02 官方给的是区间不是固定条数，长任务和云端任务更费

03 把模型名写在一个地方（配置文件或 profile），别散落在脚本里

04 模型会下线，写死的名字是几个月后静默失败的头号原因

订阅	按席位，按人算	给人用
API	按量，按 token	给程序用
额度	区间，不是固定条数	任务越重越费
模型名	只写一处	会下线

别把模型名写死在十个脚本里。

从一个人的工具，到团队的做法

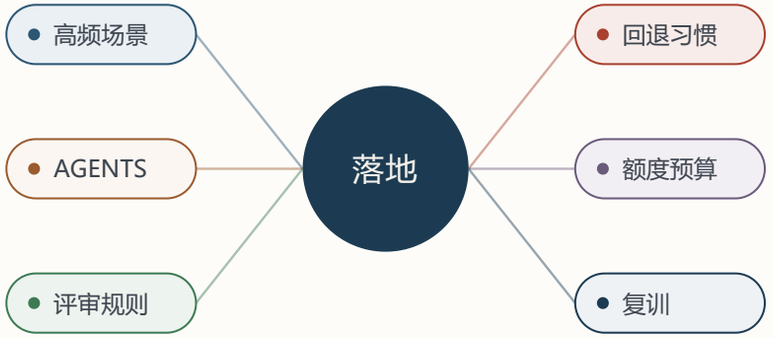
装上去只要十分钟，形成做法要改流程。

01 先挑一个每周都要做的活：改依赖、补测试、修告警

02 AGENTS.md 由最熟这个仓库的人写，不是让 AI 写

03 评审和合并的规则不变，只是多了一道自动检查

04 记下哪些任务它做得好、哪些不行，三个月后就有自己的清单



团队用的不是 Codex，是你们仓库的 AGENTS.md。

这门课之后

你应该能在自己的仓库里跑通一次完整任务，并且知道边界在哪里。

下一步：在你最熟的那个仓库，跑一次 /init。

