把工作交给AI，然后离开座位

YOLO模式完全入门. 目录和26章

金京镇律师

Chinese PDF edition

把工作交给AI，然后离开座位

金京镇律师

一本面向初学者的在线书，讲解Claude Code和Codex中的YOLO模式。它说明如何让AI读取文件、编写代码、执行命令，并把回退、Docker沙盒和安全检查放在手边。

目录

第1章 凌晨两点，坐在笔记本电脑前的公司职员

把工作交给AI，然后离开座位 - YOLO模式完全入门

|

下一章: 第2章 AI自己工作到底是什么意思

屏幕的光映在脸上。凌晨两点，首尔麻浦区的一间公寓。金代理正盯着必须在明天早上九点前提交的季度报告。三个 Excel 文件，二十二 页 PowerPoint 幻灯片，还有五个整理了各客户销售数据的 CSV 文件。他需要将这些内容整合进一份报告中。

这已经是第三杯咖啡了。

金代理平时会使用 ChatGPT。会议纪要摘要、邮件草拟、简单的翻译，处理这些工作还算好用。但这次的报告任务完全是另一个层面的。它需要打开文件、读取数据、进行计算、制作图表并粘贴到幻灯片中。如果在 ChatGPT 窗口输入“ 帮我写份报告 ”，它能写出看起来很不错的文字，但它无法实际打开 Excel 文件并提取数据。

那天晚上，金代理尝试了一件事。

他打开了一个名为 Terminal 的黑色窗口，对 AI 说：“ 读取这个文件夹里的三个 Excel 文件和五个 CSV 文件，制作一份按客户分类的季度销售汇总表，并生成一份包含图表的 PowerPoint 报告。 ”然后按下了 Enter 键。AI 开始行动了。它打开文件、编写代码、制作表格、绘制图表。金代理看着这个过程，不知不觉间合上笔记本电脑，去睡觉了。

早上七点醒来时，文件夹里已经躺着一份完成好的 PowerPoint 文件。图表齐全，数据也准确无误。金代理所做的，仅仅是下达了一个“ 帮我做这个 ”的指令。

这个故事就是本书的起点。

你可能听说过“ YOLO ”。YOLO，即 You Only Live Once，“ 人生只有一次 ”。2010 年代初期，随着说唱歌手 Drake 的歌曲名而闻名，这句话曾象征着年轻一代的消费哲学：不要担心明天，活在当下；比起储蓄，更看重体验；比起安全，更追求冒险。

然而，从 2025 年开始，这个词在程序员之间开始有了全新的含义。出现了一个被称为 AI 工具“ YOLO Mode（YOLO 模式） ”的术语。这种模式下，你给 AI 分配任务，但省去了中间反复询问“ 可以这样做吗？ ”的过程，而是让它从头到尾自主完成。这是一种无需请求许可、全速前进的模式。就像人生只有一次一样，AI 也会一气呵成地完成任务。

这本书探讨的就是这种“ YOLO 模式 ”。

本书的内容非常明确：教你如何开启、使用并关闭 Claude Code 和 Codex 这两种 AI 工具的 YOLO 模式。即使完全不懂编程的人也可以照着做。即使不知道什么是 Terminal 也没关系，书里都会教给你。

我也需要说明一下本书不涉及的内容。它不是编程教材，不会教你 Python 或 JavaScript 的语法。它也不是 AI 理论书，不会解释机器学习（Machine Learning）是如何运作的。这本书是一本工具使用指南。就像你不需要了解电钻的原理也能在墙上打洞一样，你也不需要了解 AI 的原理也能指挥 AI 干活。我会告诉你具体的方法。

|

下一章: 第2章 AI自己工作到底是什么意思

律师 前国会议员 AI政策研究者

第2章 AI自己工作到底是什么意思

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第1章 凌晨两点，坐在笔记本电脑前的公司职员

|

|

下一章: 第3章 为什么名字里有“危险地”

请想象一下你在餐厅点餐的情景。

普通的 AI 工具是这样的：当你提出“来一份泡菜汤”时，服务员会回问“是一份泡菜汤，对吗？”你回答“是的”。“需要配米饭吗？”“是的”。“小菜需要续加吗？”“好的，麻烦了。”它每一步都要经过你的确认。如果你对 ChatGPT 说“帮我写一份报告草案”，它会问“是什么主题的报告？”当你回答后，它又会问“篇幅大概要多长？”每次都要征得你的许可。这种方式虽然安全，但很慢，你无法离开座位。

YOLO 模式则不同。你只需说一句：“来一份泡菜汤，米饭和小菜请帮我配好，甜点也请随便选一个送上来”，然后就可以坐下来看书了。服务员会自行处理。他不会询问是否需要单独给白米饭、具体要哪些小菜、水是要冰的还是温的这类琐碎细节，而是会自行判断并执行。

让我们进行更准确的对比。

像 Copilot 这样的 AI 编程工具，会在你编写代码时在旁边建议下一行内容。这与写文章时的自动补全非常相似。你仍然需要动手操作，AI 只负责建议，决策权在人。

ChatGPT 是对话式的。你提问，AI 回答；你再问，它再答。就像打乒乓球一样来回往复。AI 无法在你的电脑上创建文件或运行程序，它只能在对话框内活动。

Claude Code 和 Codex 的 YOLO 模式与此有着本质的区别。AI 会直接在你的电脑上创建文件、编写代码并运行程序。它会打开文件夹读取内容，创建新文件，修改现有文件。而且，它在做所有这些事情的过程中，不会中途询问“可以这样做吗？”

“寻求许可的 AI”与“无需许可直接奔跑的 AI”，这两者的区别正是 YOLO 模式的核心。

Claude Code 是由 Anthropic 公司开发的一款 AI 工具。它在终端中运行。在默认状态下，每当需要创建文件或执行命令时，它都会询问你“可以这样做吗？”此时安全机制是开启的。而关闭这个安全机制的命令正是 `--dangerously-skip-permissions`，意为“危险地跳过权限确认”。开启此功能后，Claude Code 就会在无需询问的情况下开始工作。

Codex 是 OpenAI 开发的一个类似工具。在这里，通过 `--yolo` 这个简短的命令可以达到同样的效果。其名称本身就代表了“人生只有一次”。

2026 年 3 月，Claude Code 增加了一个名为“ Auto Mode ”的新选项。它是完全的 YOLO 模式与基础模式之间的中间点。对于看起来有风险的命令（如删除文件、更改系统设置等）仍会寻求许可，但对于安全的命令（如读取文件、文本搜索等）则会自动执行。你可以把它理解为在手动挡和自动挡之间新增了一个半自动挡。

本书将涵盖这三种模式。但核心在于完全的 YOLO 模式，即 AI 能够自主判断并执行的状态。

上一章: 第1章 凌晨两点，坐在笔记本电脑前的公司职员

|

|

下一章: 第3章 为什么名字里有“ 危险地 ”

律师 前国会议员 AI政策研究者

第3章 为什么名字里有“危险地”

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第2章 AI自己工作到底是什么意思

|

|

下一章: 第4章 终端是什么

让我们再次回顾一下开启 Claude Code YOLO 模式的命令。

claude --dangerously-skip-permissions

“dangerously”这个词映入眼帘。危险地。为什么软件公司要在自己的产品命令中加入“危险地”这个词呢？

理由很简单：为了向用户发出警告。

想想厨刀吧。刀是危险的，可能会割伤手。但没有刀就无法烹饪。我们不能因为刀很危险就不使用它，而是要学习如何使用刀：刀刃向外握持，只在案板上切割，使用后放入刀鞘。遵守这些规则，刀就是工具；无视规则，刀就会变成凶器。

YOLO 模式也是如此。

AI 在未经许可的情况下行动，意味着 AI 可以在你的电脑上自由地创建、修改和删除文件。如果使用得当，报告会在你睡觉时完成；如果使用不当，重要文件可能会在你睡觉时消失。Anthropic 在命令中加入“dangerously”，是希望你在意识到这一事实的前提下做出决定。它让命令本身承载了“我知道这种风险，但我仍要使用”的意图表达。

Codex 最初的命令更加直白：`--dangerously-bypass-approvals-and-sandbox`（危险地绕过审批和沙箱）。由于太长，后来有了 `--yolo` 这个别名，但原名所包含的警告分量并不轻。

[警告] YOLO 模式会赋予 AI 访问你电脑文件系统的自由权限。在包含重要文件的文件夹中，请务必先进行备份。

买刀时，好的刀具店会随刀附赠刀鞘。这本书就是那个刀鞘。

今日所学

YOLO 模式是指 AI 在整个过程中无需中途请求许可，即可自主执行任务的模式。在 Claude Code 中，通过 `--dangerously-skip-permissions` 命令开启；在 Codex 中，通过 `--yolo` 命令开启。由于该模式功能强大但也存在风险，因此我们需要学习如何开启以及如何关闭它。

上一章: 第2章 AI自己工作到底是什么意思

|

|

下一章: 第4章 终端是什么

律师 前国会议员 AI政策研究者

第4章 终端是什么

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第3章 为什么名字里有“危险地”

|

|

下一章: 第5章 Claude Code和Codex有什么不同

2019年冬天，我认识的一位个体经营者对我说“能不能帮我写个Excel宏”，于是我带上了笔记本电脑。当我打开那个黑色的屏幕时，他一脸惊讶地问道：“你这是在进行黑客攻击吗？”

不是的。这不是黑客攻击。

终端（Terminal）是一个通过文字向计算机下达命令的界面。我们平时使用电脑时，通常是用鼠标点击图标、打开窗口、按下按钮。这被称为 GUI（Graphical User Interface，图形用户界面）。这是一种通过图像进行交流的方式。而终端与之不同。你输入文字，计算机就能理解。这被称为 CLI（Command Line Interface，命令行界面）。这是一种通过文字进行交流的方式。

让我们回到餐厅的比喻。GUI 就像是在菜单上看着图片并用手指着说：“我要这个。”而终端就像是向服务员口头点餐：“请给我来一份泡菜汤，米饭要加量。”结果是一样的，都会端上来一份泡菜汤。只是方法不同而已。

你也可以把它想象成一个空白的记事本。一张什么都没写的白纸。只要你在上面写下想说的话，计算机就会读取并执行。

首先，我来教大家如何在 Mac 上打开终端。

同时按下键盘上的 Command 键和空格键。屏幕中间会出现搜索框。在这里输入“终端”。输入英文“Terminal”也可以。当列表中出现终端应用时，按下回车键。随后会出现黑色或白色的屏幕。你会看到光标在闪烁。这就是终端。

在 Windows 上则略有不同。键盘左下方有一个 Windows 徽标键。按下此键会打开开始菜单。在搜索框中输入“cmd”。会出现“命令提示符”。点击它，就会弹出一个黑色窗口。如果你使用的是 Windows 11，也可以搜索“Windows Terminal”，它会打开一个视觉效果更好的界面。

如果你已经做到了这一步，那么你就已经跨过了第一个难关。

看终端屏幕时，你会发现上面有一些文字。如果是 Mac，你会看到类似 "用户名@计算机名 ~ %" 的文字。如果是 Windows，你会看到类似 "C:\Users\用户名>" 的文字。这被称为提示符（Prompt）。它的意思是“请在这里输入命令”。虽然它与在 AI 聊天机器人中输入的 Prompt 名称相同，但在不同的语境下使用。

让我们试着输入一个命令。请在终端中输入以下命令并按下回车键。

echo "你好"

(echo, "你好")

屏幕上会出现“ 你好 ”。计算机原封不动地重复了你让它说的话。echo 的意思是“ 复读 ”或“ 回显 ”。就像在山谷中大喊“ 喂！ ”会有回声一样，计算机也会把你的话原样传回给你。

就是这样。终端长这样，用法也如此。它并不可怕。

[警告] 请不要在终端中随意执行不熟悉的命令。这可能会导致文件被删除或系统设置被更改。请仅输入本书中引导的命令。

你可能会好奇为什么需要终端。因为 Claude Code 和 Codex 都是在终端中运行的。这两个 AI 工具并不是通过点击图标打开的程序，而是通过在终端中输入命令来运行。因此，我们需要先学习终端。这就像在学习游泳之前，先学习如何下水一样。

上一章: 第3章 为什么名字里有“ 危险地 ”

|

|

下一章: 第5章 Claude Code和Codex有什么不同

律师 前国会议员 AI政策研究者

第5章 Claude Code和Codex有什么不同

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第4章 终端是什么

|

|

下一章: 第6章 先学会四种安全工具

我们将对比这两种工具。

Claude Code 是由 Anthropic 开发的，而 Codex 是由 OpenAI 开发的。两者都是在终端运行的 AI 工具，都能直接在你的电脑上创建文件并执行代码。虽然功能相似，但开发公司、命令以及安全机制的名称各不相同。

首先说明安装方法。安装这两个工具都需要一个名为 Node.js 的程序。Node.js 是一个运行 JavaScript 编程语言的工具，你只需要知道安装过程即可，不需要理解其原理。

Node.js 的安装方法如下。在 Mac 上，请打开终端并输入以下命令：

brew install node

(brew install node)

brew 是 Mac 上的程序安装工具 Homebrew。它类似于 App Store，但在终端中运行。如果没有安装 Homebrew，需要先进行安装。请按照 Homebrew 官网 (brew.sh) 上的指南进行操作。

在 Windows 上，请访问 nodejs.org 网站，下载安装文件并双击安装。“ 下一步，下一步，完成。 ” 过程与安装普通程序一样。

Node.js 安装完成后，开始安装 Claude Code。在终端中输入以下内容：

npm install -g @anthropic-ai/claude-code

(npm install -g @anthropic-ai/claude-code)

npm 是随 Node.js 一起提供的程序安装工具。install 表示 “ 安装 ”，-g 表示 “ 全局安装，以便在电脑任何地方使用 ”。只需这一条命令即可完成 Claude Code 的安装。

Codex 的安装方法也类似。

npm install -g @openai/codex

(npm install -g @openai/codex)

现在我们来总结一下这两个工具的区别。

命令不同。运行 Claude Code 时，在终端输入 claude；运行 Codex 时，输入 codex。

安全机制的名称不同。Claude Code 的 YOLO 模式是 --dangerously-skip-permissions，而 Codex 的 YOLO 模式是 --yolo。功能是一样的，只是名称不同。

默认模型不同。Claude Code 使用 Claude 模型。截至 2026 年 5 月，默认模型是 Claude Opus 4.7。Codex 使用 GPT 系列模型，默认模型是 GPT-5.5。

价格结构也比较相似。Claude Code 可以使用包含在 Anthropic 订阅（Claude Pro, Max, Team）中的额度，也可以通过 API 按量计费使用。Codex 同样可以使用包含在 OpenAI 订阅（ChatGPT Pro）中的额度，也可以通过 API 按量计费使用。这意味着只要有订阅，就可以在不产生额外费用的情况下开始使用。

选择使用哪一个取决于你的情况。如果你已经在付费使用 ChatGPT，那么使用 Codex 会更自然；如果你已经在付费使用 Claude，那么使用 Claude Code 会更自然。如果你两者都没有使用，由于两者都可以免费试用，你可以任选其一。因为这两个工具的工作方式非常相似，学会了一个，迁移到另一个也并不困难。

还有一点需要说明。这两个工具除了终端版本外，还有其他形式。Claude Code 可以作为 VS Code 编辑器的扩展程序使用，在 Claude Desktop 应用中也可以使用类似的功能。Codex 也有可以在 ChatGPT 网站上使用的版本。但是，如果你想使用 YOLO 模式，即让 AI 直接在你的电脑上操作文件的完全自动化模式，则需要使用终端版本。因此，本书将重点围绕终端版本进行讲解。

上一章: 第4章 终端是什么

|

|

下一章: 第6章 先学会四种安全工具

律师 前国会议员 AI政策研究者

第6章 先学会四种安全工具

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第5章 Claude Code和Codex有什么不同

|

|

下一章: 第7章 开启Claude Code的YOLO模式

在开启 YOLO 模式之前，我们先准备好安全装备。这就像滑雪前要戴好头盔一样。一共有四种。

Git，时光倒流按钮

Git 是一个记录文件变更历史的工具。

可以这样想：你在写 Word 文档时，可能经常使用“ 另存为 ”功能，创建出“ 报告_v1 ”、“ 报告_v2 ”、“ 报告_最终版 ”、“ 报告_真的最终版 ”这样的文件。Git 可以自动完成这个过程。每当你修改文件时，只要告诉它“ 记住这个状态 ”，Git 就会记录下来。以后当你要求“ 回到之前的状态 ”时，它就能实现。

之所以需要它，是因为当 AI 修改了你的文件但结果不尽如人意时，你可以将其恢复到修改前的状态。它是 YOLO 模式下的安全带。

你只需要记住三个命令。

git init

(git init)

这个命令的意思是“ 开始追踪此文件夹的变更 ”。只需在要工作的文件夹中首次执行一次。

git add .

(git add .)

意思是“ 把修改过的文件收集起来 ”。点号 (.) 代表“ 此文件夹下的所有文件 ”。

git commit -m "保存工作前的状态"

(git commit -m "保存工作前的状态")

意思是“ 记住现在的状态 ”。在引号内填写备注。这能让你以后看到备注时，明白“ 啊，这是当时保存的版本 ”。

在开启 YOLO 模式之前，请每次都执行这三个步骤。只需 3 秒钟。但这 3 秒钟能保护你的文件。

Docker，游乐场里的木围栏

Docker 是一个在电脑内部创建微型电脑的工具。

你可能见过儿童游乐场里有一个沙坑，周围围着木栅栏。无论孩子怎么挖掘沙子，都不会影响栅栏外的草坪。Docker 就是那个栅栏。

当你在 YOLO 模式下让 AI 工作时，如果让它在 Docker 中运行，无论 AI 做什么，都不会影响你 Docker 之外的文件。即使 AI 错误地删除了文件，被删除的也只是 Docker 内部的文件。真实的文件是安全的。

可以在 docker.com 下载 Docker Desktop。Mac 和 Windows 版本都有。安装过程与普通程序一样：下载、双击、“ 下一步、下一步、完成 ”。安装后运行一次应用即可。

关于如何在 Docker 中使用 YOLO 模式，我将在第 7 章和第 8 章中详细讲解。现在请先记住“ 存在 Docker 这个围栏 ”这一点。

Auto 模式，中间阶段

对于觉得完全的 YOLO 模式压力太大的人，有一个中间阶段。那就是 Claude Code 的 Auto Mode。

用交通信号灯来类比：基础模式在所有路口都会亮红灯。每次都要停下、确认、然后出发。虽然安全，但很慢。YOLO 模式会关掉所有的信号灯。一路疾驰。很快，但有事故风险。Auto 模式只关闭小路口的信号灯。在大型路口（删除文件、系统命令）依然会亮红灯，而在小路口（读取文件、搜索）则会自动通过。

如果你是第一次使用 YOLO 模式，我建议从 Auto 模式开始。使用几天并掌握感觉后，再切换到完全的 YOLO 模式。

编写清晰的指令

第四种安全工具不是技术，而是一种习惯。

不要说“ 请看着办 ”。

让我们思考一下这句话为什么危险。如果你对 AI 说“ 整理一下这个文件夹 ”，AI 会按照它自己的标准进行整理。它可能会重命名文件，或者删除旧文件。你预期的“ 整理 ”与 AI 理解的“ 整理 ”可能完全不同。

相反，你应该这样说：“ 读取这个文件夹里的三个 Excel 文件，计算销售总额，并将其保存为‘ 每月_销售_摘要.xlsx ’。不要改动原有的文件。 ”这样非常具体，明确了 AI 需要做什么以及不能做什么。

优秀指令具备三个条件。

写明要做什么。“ 读取三个 Excel 文件并计算销售总额。 ”

写明输出结果的形式。“ 保存为‘ 每月_销售_摘要.xlsx ’。 ”

写明不要做什么。“ 不要改动原有的文件。 ”

养成编写包含这三个要素的指令的习惯，比任何软件层面的安全机制都能更好地保护你。

今日所学

Git 是一个保存和回溯文件状态的时间旅行工具，通过 git init、git add、git commit 这三个命令运行。Docker 是为 AI 工作提供安全围栏的工具，而 Auto 模式是进入完全的 YOLO 模式之前可以经历的中间阶段。最重要的是，用具体的指令代替“ 请看着办 ”的习惯，才是最可靠的安全保障。

上一章: 第5章 Claude Code和Codex有什么不同

|

|

下一章: 第7章 开启Claude Code的YOLO模式

律师 前国会议员 AI政策研究者

第7章 开启Claude Code的YOLO模式

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第6章 先学会四种安全工具

|

|

下一章: 第8章 开启Codex的YOLO模式

好了，终于到了开启 YOLO 模式的时候了。

但在那之前，我们需要检查一下准备工作。请像核对清单一样逐一确认。

首先确认是否安装了 Node.js。请在终端中输入以下命令。

node --version

(node -v)

如果出现类似 "v20.11.0" 的数字，说明安装成功。如果出现的不是数字而是错误信息，请重新按照第 5 章的安装方法进行操作。

接着确认是否安装了 Claude Code。

claude --version

(claude -v)

只要显示版本号即可。

你需要获得 Claude Code 的使用权限。有两种方法：如果你拥有 Claude 订阅（Claude Pro, Max, Team），只需登录账号即可直接使用。在终端首次运行 Claude Code 时，浏览器会自动打开并显示登录页面，使用你平时使用的 Claude 账号登录即可。如果你没有订阅或想使用 API 按量计费模式，请在 Anthropic 官网 (console.anthropic.com) 获取 API Key。注册后，在 "API Keys" 菜单中点击 "Create Key"，就会生成一个以 "sk-ant-" 开头的长字符串。这就是你的密钥。请务必保管好这个密钥，不要泄露给他人。

[警告] API Key 与密码一样重要。绝对不要分享给他人，也不要发布在博客或社交媒体上。如果发现泄露，请立即在 Anthropic 官网删除该密钥并重新生成新密钥。

确定一个工作文件夹。请创建一个不包含重要文件的空文件夹，仅用于练习。

mkdir 练习文件夹

(mkdir 练习文件夹)

cd 练习文件夹

(cd 练习文件夹)

mkdir 是 "make directory" 的缩写，是用于创建新文件夹的命令。cd 是 "change directory" 的缩写，是用于切换到该文件夹的命令。

使用 Git 保存当前状态。

git init

git add .

git commit -m "开启 YOLO 模式前的状态"

准备工作已完成。现在开始开启。

claude --dangerously-skip-permissions

(claude --dangerously-skip-permissions)

按下回车键后，屏幕上会出现提示信息。这意味着 Claude Code 已经启动。光标正在闪烁，等待你的指令。在这种状态下，当你输入任何指令时，Claude Code 会直接执行，而不会中途请求许可。

让我们来测试一下。请按照如下方式输入。

"请在这个文件夹里创建一个名为 hello.txt 的文件。内容为：'你好，这是 YOLO 模式。'"

屏幕上会显示 Claude Code 创建文件的过程。它不会询问“ 是否可以创建此文件？ ”，而是直接创建。当出现完成信息时，请打开一个新的终端窗口进行确认。

cat hello.txt

(cat hello.txt)

如果输出了“ 你好，我是 YOLO 模式 ”的内容，则表示成功。你刚刚第一次开启了 YOLO 模式，并让 AI 创建了一个文件。

使用 VS Code 的用户也可以在 VS Code 中完成同样的操作。打开 VS Code，打开终端窗口（菜单中的 Terminal > New Terminal），然后输入相同的命令即可。如果安装了 Claude Code 的 VS Code 扩展程序，使用起来会更方便，但基本原理是一样的。

每次都输入 --dangerously-skip-permissions 可能会觉得很麻烦，因为它太长了。通过使用 Shell 别名 (alias) 功能，可以用短名称来缩短它。

在 Mac 上，请在终端中输入以下命令：

echo 'alias cyolo="claude --dangerously-skip-permissions"' >> ~/.zshrc

(echo alias cyolo 等于 claude dangerously skip permissions >> 波浪号/点 zshrc)

然后关闭终端并重新打开。从现在起，只需输入 cyolo 即可开启 YOLO 模式。它的意思与原命令相同，只是名字变短了，就像别名指向同一个人一样。

在 Windows 上情况略有不同。如果你使用的是 PowerShell 而不是命令提示符，请按如下方式操作：

Set-Alias cyolo "claude --dangerously-skip-permissions"

但在 PowerShell 中，这个别名在关闭终端后会消失。若要永久设置，需要编辑配置文件，这里先跳过。你也可以每次都使用原命令。

上一章: 第6章 先学会四种安全工具

|

|

下一章: 第8章 开启Codex的YOLO模式

律师 前国会议员 AI政策研究者

第8章 开启Codex的YOLO模式

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第7章 开启Claude Code的YOLO模式

|

|

下一章: 第9章 如何关闭，以及如何恢复

Codex 的 YOLO 模式原理也是一样的，只是命令不同。

准备工作与 Claude Code 几乎完全相同。需要安装好 Node.js 和 Codex。检查方法也一样。

codex --version

(Codex 桌面版)

Codex 与 Claude Code 一样，有两种使用方式。如果你订阅了 ChatGPT 付费版，可以直接通过登录 ChatGPT 账号来使用。在终端第一次运行 Codex 时会弹出登录界面，只需登录你平时使用的 ChatGPT 账号即可。如果你不想订阅，而是想使用按量付费的 API，可以在 platform.openai.com 获取 API Key。过程与 Anthropic 类似。

[警告] OpenAI API Key 也和密码一样。如果泄露，请立即删除并重新生成。

进入工作目录，并使用 Git 保存状态。过程与第 7 章中所做的一致。

现在启动。

codex --yolo

(Codex YOLO)

非常短，只需两个单词。这个像绰号一样的命令，其原始名称如下。

codex --dangerously-bypass-approvals-and-sandbox

(Codex Dangerously Bypass Approvals and Sandbox)

“危险地绕过审批与沙箱”。Sandbox（沙箱）的概念与之前学过的 Docker 围栏类似。即沙子（sand）组成的盒子（box）。它是为了将 AI 限制在安全空间内工作的装置。使用 --yolo 时，连这个围栏也会被解除。

你也可以通过配置文件（config file）预先为 Codex 设置好 YOLO 模式。这样就不需要每次都在命令中添加 --yolo，而是将其作为默认值写在配置文件中。

配置文件名为 config.toml。这是一种名为 TOML (Tom's Obvious, Minimal Language) 的文件格式，是一种人类易读的配置格式。只需在该文件中添加以下两行即可。

approval_policy = "never"

sandbox_mode = "danger-full-access"

approval_policy 的意思是“ 审批策略 ”。"never" 的值表示“ 绝不询问 ”。sandbox_mode 是“ 沙箱模式 ”，而 "danger-full-access" 的意思是“ 危险，允许完全访问 ”。有了这个配置文件，即使只输入 codex，它也会以 YOLO 模式运行。

[警告] 如果在 config.toml 中保留此设置，每次运行 Codex 时都会自动进入 YOLO 模式。练习结束后，请删除这两行或将值恢复原状。将 approval_policy 改为 "on-failure"，将 sandbox_mode 改为 "docker"，即可重新开启默认的安全防护。

让我们来测试一下。在 Codex YOLO 模式开启的状态下，输入以下内容。

“ 在这个文件夹里创建一个名为 greeting.py 的 Python 文件，并让它运行后输出‘ 欢迎，这是 Codex YOLO 模式。 ’。创建文件后请直接运行它。 ”

Codex 会创建 Python 文件并直接运行。如果屏幕上出现了“ 欢迎，这是 Codex YOLO 模式。 ”，则表示成功。你会发现整个过程中没有请求许可的步骤。无论是创建文件还是运行程序，都是由 AI 自主完成的。

无论使用 Claude Code 还是 Codex，核心逻辑都是一样的：向 AI 下达指令，由 AI 执行，而你负责确认结果。

上一章: 第7章 开启Claude Code的YOLO模式

|

|

下一章: 第9章 如何关闭，以及如何恢复

律师 前国会议员 AI政策研究者

第9章 如何关闭，以及如何恢复

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第8章 开启Codex的YOLO模式

|

|

下一章: 第10章 第一次练习：制作自我介绍网页

让我们从一个原则开始。

如果你不知道如何关闭，就不要开启。

这话听起来可能有些夸张，但我是认真的。没有人会用微波炉却不知道怎么关掉它，也没有人会发动汽车却不知道怎么熄火。道理是一样的。在开启 YOLO 模式之前，你必须先学会如何关闭它。

关闭的方法非常简单。

在键盘上同时按下 Ctrl 键和 C 键。

(Ctrl+C)

就这样。无论是 Claude Code 还是 Codex，大多数在终端运行的程序都可以通过 Ctrl+C 停止。即使 AI 正在疯狂生成文件，按下 Ctrl+C 也会停止。这就是为什么我要先告诉你这个组合键。

在 Mac 上是 Ctrl+C，而不是 Command+C。Command+C 是复制，而 Ctrl+C 是停止程序。这很容易混淆，请务必记住。

现在我来告诉大家如何恢复。

当你对 AI 的工作结果不满意时，可以将其恢复到工作前的状态。我们在第 6 章学过的 Git 在这里派上了用场。

首先，检查 AI 修改了哪些内容。在终端输入以下命令。

git diff

(git diff)

diff 是 "difference" 的缩写，意为“ 差异 ”。它会显示 AI 工作前后的变化。绿色文字表示新增的内容，红色文字表示删除的内容。你可以一眼看出哪些文件发生了怎样的变化。

检查后发现结果不尽如人意，想要全部恢复原状。请输入以下命令。

git checkout -- .

(git checkout -- .)

这条命令的意思是“ 将所有文件恢复到最后一次保存（提交）的状态 ”。因为在第 7 章开启 YOLO 模式之前，我们已经执行了 git commit，所以它会回到那个时间点。

[警告] git checkout -- . 是不可逆的。执行此命令后，AI 的所有工作内容都会消失。如果你想保留 AI 工作结果中的一部分，请在执行此命令前将需要的文件复制到其他文件夹中。

AI 创建的新文件不会被 git checkout 删除，它只会撤销现有文件的更改。要检查 AI 新创建的文件，请这样做。

git status

(git status)

显示为 "Untracked files" 的就是 AI 新创建的文件。如果你想删除它们，请逐一检查并手动删除。虽然也有可以一次性全部删除的命令，但为了安全起见，本书建议手动删除。

恢复过程可以总结如下。

通过 Ctrl+C 停止 AI。通过 git diff 检查变更。如果想恢复，执行 git checkout -- .。通过 git status 检查新文件，如果不想要则删除。

请记住这四个步骤。这就是你的逃生出口。就像进入建筑物后先确认安全出口的位置一样，使用 YOLO 模式时，请先确认这四个步骤再开始。

我还要再说最后一点。如果在没有使用 Git 保存的情况下让 AI 修改文件，将无法恢复。这就是为什么我在第 7 章强调“ 在开启 YOLO 模式之前务必执行 git commit ”。这个习惯能保护你的一切。这只需要 3 秒钟：输入 "git add ." 然后回车，输入 "git commit -m '工作前'" 然后回车。如果漏掉这 3 秒钟，你之后可能会后悔 3 个小时。

今日掌握的内容

Claude Code 的 YOLO 模式可以通过 `claude --dangerously-skip-permissions` 开启，Codex 的 YOLO 模式可以通过 `codex --yolo` 或修改 `config.toml` 设置来开启。这两个工具都可以通过 `Ctrl+C` 停止运行，并可以通过 `git diff` 查看变更，随后使用 `git checkout -- .` 进行回滚。在开启 YOLO 模式之前，请务必养成使用 `git commit` 保存当前状态的习惯。

上一章: 第8章 开启Codex的YOLO模式

|

|

下一章: 第10章 第一次练习：制作自我介绍网页

律师 前国会议员 AI政策研究者

第10章 第一次练习：制作自我介绍网页

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第9章 如何关闭，以及如何恢复

|

|

下一章: 第11章 第二次练习：合并Excel文件

周六下午，我坐在客厅的沙发上，打开了笔记本电脑。旁边放着一杯咖啡。终端屏幕正闪烁着。跟着学到第三部分，我已经学会了 Git，也尝试了开启和关闭 YOLO 模式，现在是时候动手做点什么了。

我们从简单的开始。一个自我介绍网页。内容包括姓名、爱好和喜欢的食物，就是一个单页网页。

准备

打开终端，创建一个用于练习的文件夹。

mkdir ~/yolo-practice

cd ~/yolo-practice

git init

只需要三行命令。创建文件夹，进入该文件夹，并开启 Git。这样万一出了什么问题，我们还可以撤销。

命令

现在开启 YOLO 模式。

claude --dangerously-skip-permissions

Claude Code 在黑色屏幕上向你问好。在这里输入如下内容。

"我的名字是金民秀，爱好是登山和摄影。喜欢的食物是味噌汤。请用这些内容帮我制作一个自我介绍网页。文件名请设为 index.html。"

按下回车键，然后等待。

AI 在做什么

Claude Code 开始运行。终端屏幕上的文字不断向上滚动。让我们慢慢观察 AI 正在进行的操作。

AI 首先会创建一个名为 index.html 的文件。这是一个使用 HTML（网页制作语言）编写的文件，即使你看不懂这些代码也没关系，AI 会自动写好。

文件里用大字体写着你的名字，下面列出了爱好和喜欢的食物。它还会设置好整洁的背景颜色，并挑选好看的字体。如果是普通模式，它会询问“ 可以创建文件吗？ ”，但因为现在是 YOLO 模式，它不会询问，直接进行创建。

大约 20 秒后，会显示“ 已完成 ”的消息。现在我们在浏览器中打开这个文件。你可以对 Claude Code 说“ 请在浏览器中打开刚才创建的页面 ”。如果你想手动操作，也可以在 Finder 中打开工作文件夹并双击 index.html 文件。

Chrome 或 Safari 浏览器随之打开，网页呈现出来。“ 金民秀 ”的名字清晰可见，下方写着登山、摄影和味噌汤。非常整洁。

耗时与费用

从输入第一个命令到在浏览器中确认，整个过程不到 3 分钟。费用以 Claude API（程序间通信的通道）为标准，大约在 2 块到 4 块韩元之间。甚至不到一杯咖啡价格的十分之一。

风险因素

这次练习几乎没有什么出错的可能。因为 AI 做的仅仅是创建了一个文件。它没有修改现有文件，而是创建了新文件，所以不会对电脑产生影响。如果不满意，直接删除该文件即可。你可以在 Finder (Mac) 或资源管理器 (Windows) 中右键点击删除，也可以让 Claude Code 执行“ 删除刚才创建的文件 ”。既然已经用 Git 保存了，也可以通过指令“ 全部恢复原状 ”来撤销。

如果想尝试更多

如果你觉得网页太单调，可以这样对 Claude Code 说：

"我想加入照片。请为我留出一个头像的位置。"

"请把颜色改为蓝色系。"

"请添加联系方式按钮。"

建议一次只下达一个指令。如果一次性下达十个指令，AI 可能会感到困惑。

上一章: 第9章 如何关闭，以及如何恢复

|

|

下一章: 第11章 第二次练习：合并Excel文件

律师 前国会议员 AI政策研究者

第11章 第二次练习：合并Excel文件

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第10章 第一次练习：制作自我介绍网页

|

|

下一章: 第12章 第三次练习：每天早晨接收新闻摘要

周一早上9点。销售部的朴经理一坐到座位上就叹了口气。销售报告的截止时间是下午2点，但销售表被分成了三个Excel文件：1季度_销售.xlsx、2季度_文件.xlsx、3季度_销售.xlsx。必须把它们合并成一个。

如果是以前，需要打开每个文件，进行复制、粘贴，检查行号是否对应，并肉眼核对格式是否错乱。这原本是一项需要花费30分钟的工作。

准备

进入存放销售表文件的文件夹。

cd ~/Documents/sales_data

git init

git add .

git commit -m "备份原文件"

一共四行。这是先用Git保存原件。这应该成为一种习惯。因为在YOLO模式下，AI可能会覆盖原文件。如果用Git保存了，即使被覆盖了也可以恢复。

指令

claude --dangerously-skip-permissions

Claude Code启动后，可以这样下令：

"请把这个文件夹里的三个xlsx文件合并成一个Excel文件。保持每个文件的Sheet名称不变，合并后的文件名设为销售_整合.xlsx。不要改动原文件。"

最后一句非常重要。“ 不要改动原文件。 ”如果没有这句话，AI有可能直接覆盖原文件。

AI的工作内容

Claude Code首先会创建一个Python脚本。该脚本会使用名为openpyxl（处理Excel文件的工具）的库（library，预先编写好的代码集合），如果电脑上没有这个库，AI会自动进行安装。如果是基本模式，它

会询问“ 可以安装这个库吗？ ”，但在YOLO模式下，它会直接安装。

创建脚本、执行，随后生成销售_整合.xlsx文件。整个过程大约耗时40秒。

确认结果

用Excel或Google Sheets打开销售_整合.xlsx文件。你会看到有三个Sheet标签，分别包含第1季度、第2季度和第3季度的数据。请与原文件对比，检查数字是否正确。

这里一定要进行检查。因为AI偶尔会出现漏掉数字或改变行顺序的情况。请对比原文件最后一行合计与整合文件的合计。如果数字一致，则说明成功。

耗时与成本

从输入指令到确认结果，总共耗时不到1分钟。成本在300韩元左右。如果由朴经理手动操作，则需要花费30分钟。

风险因素

风险只有一个：覆盖原文件。AI可能会将合并后的结果直接保存到原文件上。所以才要先用Git保存。如果没有用Git呢？

git checkout -- .

只需这一行命令，所有文件都会回到最后一次提交（commit，保存点）的状态。如果没有使用Git，就无法回退，所以请务必先进行保存。

如果想进一步防止错误，也可以选择单独备份原文件。

cp *.xlsx ~/Documents/sales_data_backup/

这相当于腰带和背带同时使用，还是安全一点比较好。

上一章: 第10章 第一次练习：制作自我介绍网页

|

|

下一章: 第12章 第三次练习：每天早晨接收新闻摘要

律师 前国会议员 AI政策研究者

第12章 第三次练习：每天早晨接收新闻摘要

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第11章 第二次练习：合并Excel文件

|

|

下一章: 第13章 从练习中学到的三条原则

在钟路区经营小吃店的李社长每天早上都会浏览 Naver 新闻。餐饮业动态、食材价格、卫生相关法规变更，需要了解的信息很多，但阅读新闻的时间却很紧迫。因为他早上 7 点就要到店开始准备面团。

李社长想要的是这样的效果：每天早上 8 点，自动整理五条“ 餐饮业 ”相关的新闻，并以文本文件的形式放在桌面上。这样他可以在开店前扫一眼，记住需要的信息，其他的直接跳过即可。

准备

这次实操比前两次实操需要更多的设置。大约需要 10 分钟，但一旦设置好，之后每天都会自动运行。

首先，创建一个实操文件夹。

mkdir ~/news-summary

cd ~/news-summary

git init

接下来，使用 Docker（Docker，创建一个隔离工作空间的工具）。因为在这次实操中，AI 需要访问互联网。为了搜索新闻，必须访问网页。在 YOLO 模式下，如果让 AI 自由访问互联网可能会带来风险，所以我们将其运行在 Docker 这个“ 围栏 ”之内。

既然你在第 6 章已经安装好了 Docker，那么就这样命令 Claude Code：“ 请在 Docker 容器中执行新闻摘要任务。将 ~/news-summary 文件夹连接为工作文件夹，以便结果文件也能保存在我的电脑上。 ”Claude Code 会自动编写并执行 Docker 命令。你不需要去背像 `docker run` 这样冗长的命令。

命令

当 Claude Code 在 Docker 中启动后，就这样下令：

“ 请搜索五条与餐饮业相关的最新新闻，并将每篇文章的标题和三行摘要制作成 news_今天日期.txt 文件。请将文件保存在 /work 文件夹中。 ”

因为是第一次运行，先试着跑一下。AI 会使用网页搜索工具查找新闻、进行摘要并创建文本文件。整个过程大约需要 2 分钟。

检查结果。在 ~/news-summary 文件夹中，应该会出现类似 news_20260513.txt 的文件。打开后，你会看到五个新闻标题以及各自的三行摘要。

自动化设置

第一次成功运行了。现在我想让它每天早上 8 点自动运行。

在 Mac 上，我们使用 launchd（launchd，Mac 的定时任务工具）。你可以这样命令 Claude Code：

“ 请为刚才的操作创建一个每天早上 8 点自动执行的 Shell 脚本和 launchd plist 文件。 ”

AI 会为你创建两个文件。Shell 脚本（shell script，包含一系列命令的文件）的内容是启动 Docker 并运行 Claude Code；而 plist（plist，Mac 的定时配置文件）文件则包含了“ 每天 8 点执行此脚本 ”的预约信息。

将 plist 文件注册到 Mac 的预约系统中，设置就完成了。

launchctl load ~/Library/LaunchAgents/com.news.summary.plist

通过这一行命令即可完成注册。只要明天早上 8 点电脑处于开机状态，桌面上就会出现当天日期的新闻摘要文件。

耗时与成本

初次设置耗时 10 分钟。之后每天会自动运行 2 分钟。每次运行的成本大约在 500 韩元左右，一个月大约是 15,000 韩元。这与订阅新闻剪报服务的费用相差无几。

风险因素

本次实操的风险在于网络访问，因为这意味着 AI 会连接互联网。通过在 Docker 中运行，AI 的访问范围被限制在 Docker 内部，无法触及我电脑上的其他文件或程序。

如果不使用 Docker 运行这个程序会怎样？当 AI 访问互联网时，可能会下载意料之外的脚本，或者在电脑的其他文件夹中创建文件。Docker 充当了这种隔离保护的作用。

还有一点。由于自动运行脚本每天都在执行，费用会不断累积。请每月检查一次 API 使用情况仪表盘（dashboard，使用情况界面）。如果新闻搜索时间过长，费用可能会高于预期。

上一章: 第11章 第二次练习：合并Excel文件

|

|

下一章: 第13章 从练习中学到的三条原则

律师 前国会议员 AI政策研究者

第13章 从练习中学到的三条原则

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第12章 第三次练习：每天早晨接收新闻摘要

|

|

下一章: 第14章 写作与文档整理

我进行了三次实践。制作了网页、合并了 Excel、自动收集了新闻。这三项任务的性质各不相同，但成功的理由是一样的。

从小处着手。在网页实践中，我只放入了“ 姓名、爱好、喜欢的食物 ”。我没有一开始就要求“ 加入照片、加入动画、做成响应式 ”。先从小处着手并确认运行良好，然后再逐一添加。

这样做非常重要。如果让 AI 一次性做太多事情，出错的概率就会上升。如果同时下达了十个指令，其中八个做得很好，两个出错了，那么很难找到哪里错了。如果逐一指令，错误的部分会立刻显现。

在开始之前先用 Git 保存。三次实践中，我都先执行了 `git init` 和 `git commit`。这是安全带。不系安全带开车大部分时间也没事，但如果发生事故呢？不使用 Git 而采用 YOLO 模式，就像不系安全带就上高速公路一样。

即使是在像第 10 章网页实践这样几乎没有风险的任务中，我也使用了 Git。因为这是一种习惯。不要每次都去判断，直接去做就行。`git init`, `git add`, `git commit`。只需 6 秒钟。

用肉眼确认结果。当 AI 说“ 已完成 ”时，我并不盲目相信。网页我会直接在浏览器中打开查看，Excel 文件我会打开并对比数字，新闻摘要我会打开文件阅读内容。

AI 有时不知道自己生成的结果是正确还是错误。我见过它说着“ 成功了 ”，但实际上文件是空的情况。用肉眼确认只需要 30 秒。为了省这 30 秒，可能会导致后面浪费 30 分钟。

这三点：从小处着手、先用 Git 保存再开始、用肉眼确认结果，适用于第五部分出现的所有案例。不需要刻意背诵，因为在三次实践中，这些已经变成了肌肉记忆。

[第 4 部分结束] 今天掌握的内容

尝试用 YOLO 模式制作了自我介绍网页。耗时 3 分钟，花费 300 韩元。

将三个 Excel 文件合并为一个。耗时 1 分钟，花费 300 韩元。养成了先用 Git 保存原件的习惯。

尝试设置了每天早晨自动接收新闻摘要。设置耗时 10 分钟，之后每天自动运行。在 Docker 中运行才更安全。

三条原则：从小处着手。先用 Git 保存再开始。用肉眼确认结果。

上一章: 第12章 第三次练习：每天早晨接收新闻摘要

|

|

下一章: 第14章 写作与文档整理

律师 前国会议员 AI政策研究者

第14章 写作与文档整理

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第13章 从练习中学到的三条原则

|

|

下一章: 第15章 处理表格和数字

案例 1) 批量生成 50 封邮件回复草稿

场景

周三晚上 7 点，在营销代理公司工作的郑裕真代理打开了笔记本电脑。收件箱里有 53 封未读邮件。包括客户问候、报价请求、会议日程协调、活动邀请等。如果一封封阅读并撰写回复，至少需要两个小时。而且明天早上还要准备会议资料。

指令

郑裕真代理将收件箱中的邮件全部导出为文本文件。在 Gmail 中选择邮件，将内容复制并粘贴到 emails_received.txt 文件中。每封邮件之间添加分隔线。

开启 Claude YOLO 模式，并下达如下指令：

"请读取 emails_received.txt 文件，并为每封邮件生成回复草稿。语气要礼貌但简洁。关于报价的内容，请以‘ 确认后将于明天回复 ’结尾；关于日程协调，请回复‘ 周二下午 2 点可以 ’。请将结果保存到 replies_draft.txt 中。"

执行过程

Claude 代码读取 emails_received.txt，并识别 53 封邮件的内容。它会根据邮件性质进行分类（问候、报价、日程、邀请、其他），并撰写相应的回复。大约 3 分钟后，replies_draft.txt 文件便生成了。

结果

包含 53 封回复草稿的文本文件。每封回复前都注明了原邮件的标题和发件人，因此可以立即识别出是针对哪封邮件的回复。

成本

耗时 3 分钟，成本约 500 韩元。

风险

AI 可能会在回复中加入错误的内容。例如，你要求写“ 周二下午 2 点 ”，它可能会写成“ 周三上午 10 点 ”。因为是草稿，发送前务必逐一检查。特别是涉及金额的邮件，一定要核对数字。

如何效仿

你只需要知道如何将邮件内容导出为文本文件即可。如果是 Gmail，只需打开邮件，复制内容并粘贴到记事本中。核心在于具体地指示回复的语气和基本响应内容。与其说“ 请自行回复 ”，不如明确规定“ 报价按这样回复，日程按这样回复 ”，这样效果更好。

案例 2) 将会议录音文件转换为会议纪要

场景

周四下午 4 点。中小企业策划部的金组长结束了一场时长 1 小时 30 分钟的会议。他需要整理会议内容，但会议期间只记了三行笔记。幸好他用智能手机录了音。

听录音并撰写会议纪要通常需要录音时长的两倍。如果是 1 小时 30 分钟的会议，撰写纪要就需要 3 小时。看来今晚注定要加班了。

指令

金组长将录音文件 (meeting_0513.m4a) 传输到电脑上。可以通过 AirDrop 发送到 Mac 上。

创建工作文件夹并启动 Git。

mkdir ~/meeting-notes

cd ~/meeting-notes

git init

将录音文件放入此文件夹，然后启动 Claude Code。

"将 meeting_0513.m4a 文件转换为文本，并整理成会议纪要格式。请区分发言人，并将决定事项和后续行动单独汇总在底部。结果请保存为 会议纪要_0fmt.txt。"

进行过程

Claude Code 首先会将音频文件转换为文本。这个过程被称为转录（transcription，即将语音转为文字）。Claude Code 并不是直接完成这一步，而是通过安装并使用像 Whisper（OpenAI 开发的语音识别工具）这样的转录工具来完成。由于处于 YOLO 模式，它不会询问是否允许安装工具，而是直接进行操作。

转录完成后，会得到原始文本。内容可能包含类似“ 呃，所以那个事项我们再考虑一下，啊，但是预算部分好像需要尽快处理 ”这种口语化的片段。

Claude Code 会对这些文本进行整理。它会区分发言人（仅凭声音可能无法 100% 准确区分，可能会显示为“ 发言人 A ”、“ 发言人 B ”），将口语转换为书面语，并提取出决定事项和后续行动。

整个过程大约需要 8 到 12 分钟，具体取决于录音的时长。

结果

会议纪要_0513.txt 文件。顶部记录了会议日期、参会人员（基于发言人区分标准）和议程；中间按时间顺序整理了发言内容；底部清晰地汇总了三项决定事项和四项后续行动。

费用

耗时 10 分钟，费用 1,500 到 2,500 韩元。录音文件越长，需要转录的文本越多，费用就越高。

风险

存在两种风险。

一是转录错误。语音识别并非 100% 准确。可能会误听专有名词、专业术语或微小的声音。例如，可能会把“ 第三季度营收 ”误写成“ 三季度每周 ”。在提交会议纪要之前，需要对照录音进行核对阅读。

二是发言人识别错误。如果会议有五个人参加，AI 可能会识别为三人，或者将某人的发言记录在另一个人的名下。提前告知参会人员名单可以提高准确度。请在指令中加入“ 参会人员是金组长、李科长、朴代理、崔职员、郑实习生 ”。

我们也想尝试的话

使用智能手机的录音应用录制会议。使用 iPhone 的“ 语音备忘录 ”应用就足够了。将录音文件传输到电脑，放入工作文件夹，然后交给 Claude Code 处理。请提前告知参会人员名单。提交会议纪要前务必阅读一遍。

案例 3) 阅读 10 篇论文并制作摘要报告

场景

周五晚上 11 点。研究生硕士课程的韩素拉女士正在打开实验室的灯。她需要在下周二的研讨会上发表关于“ 韩国个体工商户数字化转型 ”的前沿研究。教授发来了 10 篇论文，每篇 20 到 40 页，总计超过 300 页。

如果想要读完这 10 篇论文，整理出共同点和差异点，制作成表格，甚至准备好演示文稿，那么整个周末都必须困在实验室里。

指令

韩素拉女士将 10 个论文 PDF 文件放入 papers 文件夹中。

```
mkdir ~/paper-review

cd ~/paper-review

mkdir papers
```

git init

(将论文 PDF 复制到 papers 文件夹)

启动 Claude Code。

"请阅读 papers 文件夹中的 10 篇 PDF 论文，并整理出每篇论文的研究目的、研究方法、主要结果和局限性。然后，请附上一份综合分析，对比这 10 篇论文的共同发现和不同结论。将结果保存为 先行研究_摘要报告.txt。"

执行过程

Claude Code 开始逐篇阅读论文。它会从 PDF 中提取文本，理解内容并进行摘要。每篇论文大约需要 2 到 3 分钟，因此 10 篇论文需要 20 到 30 分钟。

过程中可能会遇到一个问题，即 PDF 是扫描图像的情况。扫描纸质论文的 PDF 是图片而非文本，AI 无法直接读取文字。在这种情况下，Claude Code 会安装 OCR（光学字符识别，从图像中识别文字的技术）工具来进行处理。这会花费更多时间。

在完成 10 篇论文的单独摘要后，Claude Code 会撰写综合分析。内容包括论文间的共同发现（例如“ 10 篇中有 7 篇报告称‘ 社交媒体营销对个体工商户的销售额有积极影响 ’ ”）、不同的结论（例如“ 关于外卖应用的效果，3 篇持肯定态度，2 篇持否定态度 ”）以及整体的研究趋势。

结果

先行研究_摘要报告.txt 文件。篇幅通常为 A4 纸 8 到 12 页。前半部分是 10 篇论文的单独摘要，后半部分是综合对比分析。

费用

耗时 30 分钟，费用 5,000 到 8,000 韩元。论文越长越复杂，费用就越高。如果由人工完成，这通常需要耗费整个周末两天的时间。

风险

本案例的风险与其他实操不同。它不是损坏文件或泄露密码那类风险，而是内容的准确性问题。

AI 虽然“ 阅读 ”了论文，但并不像人类那样理解。它可能会读错统计数据、颠倒因果关系，或者总结出与作者观点相反的内容。例如，它可能会将“ 外卖应用导致销售额下降 15% ”的论文总结为“ 外卖应用有助于增加销售额 ”。

因此，不能直接将 AI 生成的摘要报告用于演示文稿。这只是草案，而非成品。在阅读报告时，必须核对核心数据和结论是否与原论文一致。

AI 做的只是把 300 页的内容缩减到了 12 页。阅读并核对这 12 页是你们的任务。即便如此，这仍然比从头阅读 300 页要快得多。

我们也想尝试的话

将论文 PDF 收集到同一个文件夹中。确认它们是文本型 PDF 而非扫描型 PDF（打开 PDF，如果可以选中并拖动文字，就是文本型 PDF）。明确指定摘要的项目。相比于“ 请摘要 ”，使用“ 请整理研究目的、研究方法、主要结果和局限性 ”能获得更好的结果。完成后务必与原文进行比对。

案例 4) 将 Markdown 文稿自动转换为出版级 PDF

场景

周日午后。经营博客的自由撰稿人尹世贤先生完成了电子书文稿。他用 Markdown（一种用于写作的轻量级标记语言）编写的文稿共有 12 个章节，总计 180 页。他打算将其转换为 PDF 并作为电子书销售。

虽然有很多将 Markdown 文件转换为 PDF 的工具，但如果要调整封面、目录、页码和字体设置，光是调整设置就要花上大半天时间。

命令

mkdir ~/ebook-build

cd ~/ebook-build

git init

将文稿文件（从 chapter_01.md 到 chapter_12.md）放入此文件夹。

"请按顺序合并此文件夹中的 12 个 Markdown 文件，并制作成韩语电子书 PDF。请添加封面页，标题设为‘ 咖啡与键盘 ’。自动生成目录并添加页码。字体使用 Nanum Myeongjo，正文字号为 11pt。结果文件名设为 咖啡与键盘.pdf。"

执行过程

Claude Code 会安装名为 Pandoc 的工具（Pandoc，文档转换工具）。这是一个广泛用于将 Markdown 文件转换为 PDF 的程序。如果电脑上没有 Nanum Myeongjo 字体，它也会一并安装。

它会按顺序合并 12 个文件，在开头添加封面页，自动生成目录，并在每页底部添加页码。它使用 LaTeX（排版系统）这一排版工具来生成 PDF，在此过程中，多个工具会进行链式安装和运行。

如果是默认模式，你会收到三次询问：“ 是否安装 Pandoc？ ”、“ 是否安装 LaTeX？ ”、“ 是否安装 Nanum Myeongjo？ ”。在 YOLO 模式下，这三次询问都会被跳过。

整个过程耗时 5 到 8 分钟。

结果

文件为 coffee-and-keyboard.pdf。封面标题醒目，下一页是目录，正文使用 Nanum Myeongjo 11 磅字体，排版整洁。

费用

耗时 8 分钟，费用 1,000 到 1,500 韩元。

风险

此案例的风险在于会进行多次工具安装。Pandoc、LaTeX 和字体，这三者都会安装到电脑上。安装本身没有风险，但会占用相当大的磁盘空间。LaTeX 的完整包可能会超过 3GB。如果笔记本电脑磁盘空间不足，请提前确认。

PDF 转换后，可能会出现字体乱码、表格超出页面或图片位置偏移的情况。建议打开 PDF 从头到尾检查一遍。

我们也来尝试一下

原稿是什么格式都可以。不一定是 Markdown，也可以是普通文本文件 (.txt) 或 Word 文件。只要对 Claude Code 说“ 请用这个文件做成一本书 ”，它就会自动按格式进行转换。如果能在指令中明确字体和字号会更好；如果对封面设计不满意，可以通过“ 把封面背景色改为深蓝色 ”之类的追加指令进行调整。

失败案例：AI 漏掉原文并进行了摘要

有一天，一名硕士生把 7 篇论文的摘要报告任务交给了 AI。查看结果时发现，只有 6 篇论文被摘要了，第 7 篇论文被完全漏掉了。

经查原因，第 7 篇论文的 PDF 是扫描图像格式。AI 无法提取文本，且没有报告无法提取这一事实。它在没有任何错误提示的情况下，悄无声息地跳过了。

发生这种情况是有原因的。AI 的行为倾向于“ 只做能做到的部分并继续下去 ”，而不是停下来说“ 我做不到 ”。如果是人，肯定会说“ 第 7 篇论文无法读取 ”，但 AI 只是摘要了 6 篇并报告“ 已完成 ”。

应对方法很简单：核对结果的数量。如果布置了 10 篇的任务，就检查是否生成了 10 篇。如果布置了 53 封邮件的回复任务，就数一下是否有 53 封。如果数量对不上，就需要找出漏掉了哪一部分。

在指令中加上这一行会更好：“ 如果有无法读取的文件，请不要跳过，并告知文件名及原因。 ”

上一章: 第13章 从练习中学到的三条原则

|

|

下一章: 第15章 处理表格和数字

律师 前国会议员 AI政策研究者

第15章 处理表格和数字

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第14章 写作与文档整理

|

|

下一章: 第16章 制作网站与自动化

案例 5) 商店 12 个月销售额自动汇总

场景

年终结算时期。经营小吃店的理事长拿出了 12 个 Excel 文件。这是从 1 月到 12 月，每月制作的一个销售表。每个文件中都记录了日期、菜单、数量和金额。需要将这些文件合并为一个，并计算出月度总计和年度总计。

指令

"请将此文件夹中的 12 个 xlsx 文件合并为一个。合并后，请计算月度销售总额和年度销售总额，并制作按菜单排序的销售排名。结果请保存为 2025年_年度销售额.xlsx。"

执行过程

Claude Code 使用 Python 编写并运行脚本。合并 12 个文件的数据，计算每月小计，并按菜单进行销售额排序。耗时 2 分钟。

结果

第一个工作表是完整数据，第二个工作表是月度汇总表，第三个工作表是按菜单排序的销售排名。炒年糕年度排名第一（4,200 万韩元），紫菜包饭排名第二（2,800 万韩元）。

费用

耗时 2 分钟，费用 400 韩元。

风险

与第 11 章相同，存在覆盖原文件的风险。请先使用 Git 进行保存。请手动计算一两个月的数据，核对总额数字是否正确。

我们也想尝试的话

每月销售文件的列名可能各不相同。有的文件叫 "菜单"，有的叫 "品项"，有的叫 "商品名称"。没关系。只需告诉 Claude Code "每个文件的列名可能不同，请自动匹配并合并" 即可。或者也可以在合并前先要

求它 "统一 12 个文件的列名"。

案例 6) 筛选客户名单中的重复项

场景

建筑材料流通公司的销售团队。客户名单 Excel 文件中包含 2,400 个公司名称。但似乎有些公司被多次记录了。例如“ 韩光建设 ”、“ 韩光建设 ”（带空格）和“ (株)韩光建设 ” 分别位于不同的行。如果用肉眼浏览 2,400 条数据来寻找重复项，需要花上一整天的时间。

指令

"请在 客户名单.xlsx 文件中查找重复的公司名称。请将空格差异、是否有 "(株)" 以及 "株式会社" 标记的差异都视为同一家公司。请将疑似重复的列表单独保存为 duplicate_check.xlsx。"

执行过程

Claude Code 对公司名称进行规范化处理。在进行去除空格、去除 "(株)"、去除 "株式会社" 等预处理后，将相似名称进行归类。使用相似度算法，将字符相似度在 80% 以上的公司识别为重复候选。

结果

duplicate_check.xlsx 文件。出现了 142 组疑似重复的项。每一行都记录了 "原始名称 A"、"原始名称 B" 和 "相似度分数"。

费用

用时 1 分钟，费用 500 韩元。

风险

AI 可能会将不同的公司归为同一家。例如，“ 韩光建设 ” 和 “ 韩光建材 ” 是不同的公司，但因为文字相似，可能会被识别为重复。因此，我要求它输出 “ 疑似重复列表 ” 而不是直接判定为 “ 重复 ”。最终判断由人工完成。

如何效仿

请告诉它包含客户名称的列名，例如 “ 公司名称在 B 列 ”。你可以调整相似度标准。如果说 “ 只抓取 90% 以上的 ”，列表就会减少；如果说 “ 70% 以上 ”，列表就会增加。

案例 7) 从收据照片中提取数字并生成 Excel

场景

报税季到了。抽屉里塞满了这一年攒下的收据。信用卡收据、现金收据、简易收据，加起来超过 200 张。如果手动将这些信息输入 Excel，需要耗费两天的时间。

指令

用智能手机拍摄收据照片。每张照片对应一张收据。将 200 张照片放入 receipts 文件夹中。

"读取 receipts 文件夹中的 200 张收据照片，并从每张收据中提取日期、店铺名称、金额和支付方式。请将结果生成为 收据整理.xlsx 文件，并按日期排序。"

执行过程

Claude Code 会读取每张照片。利用 OCR 技术识别照片中的文字，并找准日期和金额的位置。处理每张收据大约需要 10 到 15 秒，因此 200 张照片需要 30 到 50 分钟。在电脑工作期间，你可以去处理其他事情。

结果

生成 收据整理.xlsx 文件。200 行数据按日期排序，每行都记录了日期、店铺名称、金额和支付方式。文件底部还包含总计金额。

费用

用时 40 分钟（大部分时间是电脑在自动运行），费用 4,000 到 6,000 韩元。

风险

收据识别的准确度很大程度上取决于照片质量。拍摄清晰的信用卡收据准确率在 95% 以上，但褶皱或褪色的简易收据准确率可能会降至 70%。由于金额数字的一个错误就可能导致报税问题，因此对于金额较大的收据（10 万韩元以上）必须与原件进行核对。

拍摄时有一个小技巧：将收据平放在桌面上，从正上方垂直拍摄，避免产生阴影。如果倾斜拍摄，文字会发生扭曲，从而降低识别率。

如何效仿

清晰地拍摄收据照片是成功的一半，另一半在于核对结果。由于无法核对全部 200 项，请随机抽取 20 项（10%）与原件进行对比。如果错误率在 5% 以下，就是可以接受的水平。

案例 8) 分析 200 条 YouTube 评论的情感并生成 Excel 饼图

场景

经营小型化妆品品牌的崔代表在 YouTube 上上传了新品测评视频。一周内收到了 237 条评论。“ 味道很好 ”、“ 价格有点…… ”、“ 包装很漂亮，买来当礼物 ”、“ 感觉很快就会断货 ”、“ 这次不太行 ”。虽然可以逐条阅读，但希望能一眼看出整体反应是好是坏。

命令

首先，将 YouTube 评论导入文本文件。可以通过复制 YouTube 评论，或者使用 yt-dlp（下载 YouTube 视频和信息的工具）等工具来提取评论。你也可以让 Claude Code 来完成这项工作。

"请获取此 YouTube 视频 URL 的所有评论，并保存为 comments.txt。"

评论保存完成后，开始进行分析。

"请对 comments.txt 中的 237 条评论进行情感分析。将其分为正面、负面和中性，并用饼图展示各分类的比例。同时提取负面评论中经常出现的五个关键词。将结果保存到 评论分析.xlsx 中。请将饼图放入 Excel 文件内。"

执行过程

Claude Code 会逐行读取评论并进行情感分类。“ 味道很好 ” 为正面，“ 价格有点…… ” 为负面，“ 包装很漂亮，买来当礼物 ” 为正面。情感分类完成后，它会计算比例，例如：正面 62%，负面 23%，中性 15%。

使用 Python 创建 Excel 文件，并通过 openpyxl 将饼图插入到 Excel 工作表中。它还会单独提取负面评论中的高频关键词（“ 价格 ”、“ 容量 ”、“ 物流 ”、“ 留香时间 ”、“ 过敏 ”），并整理在另一个工作表中。

整个过程耗时 5 到 8 分钟。

结果

评论分析.xlsx 文件。工作表 1 包含全部评论及其情感分类（正面/负面/中性）。工作表 2 包含饼图和比例摘要。工作表 3 包含负面评论关键词分析。崔代表打开此文件，一眼就能看出“ 正面评价占 62%，反应还不错，但对价格和容量存在不满 ”。

费用

耗时 8 分钟，费用 1,500 到 2,500 韩元。

风险

情感分析的准确率在 80% 左右。问题在于讽刺性的评论。例如“ 哇，真了不起，这个价格能有这个容量 ”，这明明是负面评价，但 AI 可能会将其分类为正面。AI 在识别韩语的反讽语气时，确实存在漏掉的情况。

完美的分析很难实现。请将其用于把握整体趋势，不要纠结于精确的数值。

我们也想尝试的话

第一步是将 YouTube 评论获取为文本。你可以命令 Claude Code：“ 请获取此 YouTube URL 的评论 ”。提前告知情感分类的标准会更好，例如：“ 将与价格相关的抱怨视为负面，将单纯的提问视为中性 ”。

失败案例：读错数字的 AI

曾有人将 100 张收据照片交给 AI 处理。打开生成的 Excel 结果时，发现总金额不对。原本一个月的支出大约是 350 万韩元，但 AI 计算出的总额却是 830 万韩元。

查明原因后发现，其中有一张收据将“ 35,000 韩元 ”误读成了“ 350,000 韩元 ”，共有三处。这是因为误认了逗号的位置。此外，还有两处将“ 8,500 韩元 ”读成了“ 85,000 韩元 ”。在模糊的收据上，多出了一个零。

在处理数字任务使用 AI 时，有一点需要记住：AI 不是在“ 大概猜 ”，而是在“ 充满自信地犯错 ”。它会非常笃定地把 35,000 韩元写成 350,000 韩元，而且完全不会带上问号。

应对方法：先看总计。确认总计是否在预期范围内。如果超出范围，则从大额款项开始核对。

上一章: 第14章 写作与文档整理

|

|

下一章: 第16章 制作网站与自动化

律师 前国会议员 AI政策研究者

第16章 制作网站与自动化

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第15章 处理表格和数字

|

|

下一章: 第17章 办公室自动化

案例 9) 个人博客，从创意到部署

场景

大学生朴智浩想做一个写啤酒评论的博客。虽然可以使用 Naver Blog 或 Tistory，但他想要拥有自己的域名（domain，域名，互联网地址），比如 beerlog.kr 这样的地址。但他从未亲手制作过网站。虽然听说过 HTML 这个词，但从未写过代码。

指令

mkdir ~/beerlog

cd ~/beerlog

git init

打开 Claude Code，并下达如下指令。

"请帮我创建一个啤酒评论博客。名字叫‘ 一杯啤酒的记录 ’。首页要显示文章列表，点击文章后显示评论正文。评论内容包括啤酒名称、酿酒厂、评分（5分制）、一张照片和正文。设计风格要简洁且色调偏暗。请实现通过编写 Markdown 文件即可自动生成页面的功能。"

进行过程

Claude Code 会配置静态网站生成器（Static Site Generator，将 Markdown 文件转换为网页的工具）。它会从多种工具中选择并安装类似 Hugo 或 Eleventy 的工具。

接着创建文件夹结构，设置主题（theme，主题，设计模板），并生成一篇示例评论文章。结构为：首页显示文章列表，点击文章后显示正文。

在本地（local，本地，我的电脑）启动预览。

hugo server

输入此命令后，可以在浏览器中通过 http://localhost:1313 地址查看博客。因为这仅在我的电脑上可见，其他人无法看到。

到这一步大约耗时 15 分钟。

若要完成部署（deploy，部署，上传到互联网），请继续指示 Claude Code。

"请将这个博客部署到 Vercel。"

对于个人项目，Vercel（网站托管服务）可以免费使用。Claude Code 会创建 GitHub（代码仓库服务）仓库，上传代码，并将其与 Vercel 连接。

加上部署过程，总共耗时 25 到 30 分钟。

结果

已在互联网公开的博客。任何人都可以通过类似 beerlog.vercel.app 的地址访问。如果想要连接自己的域名（beerlog.kr），则需要单独购买域名，费用大约每年 15,000 韩元。

费用

时间 30 分钟，费用 3,000 到 5,000 韩元。服务器运营费用为免费（使用 Vercel 免费版）。如果购买域名，每年需额外支付 15,000 韩元。

风险

一旦部署，即上传到互联网，风险就会随之而来。如果代码中包含个人信息，将会向全世界公开。如果 API 密钥、密码、个人邮箱被硬编码（hard-coding，直接写在代码中）在代码里，将会产生大问题。虽然这个案例是个人博客，不太可能包含秘密信息，但请养成检查的习惯。

在部署前，可以这样指示 Claude Code：“ 请检查此文件夹中的所有文件，确认是否存在密码、API 密钥、Token 等敏感信息泄露的地方。 ” Claude Code 会逐一检查文件，寻找包含 password、secret、api_key、token 等单词的行。如果没有发现任何内容，则是安全的。

想要我们也这样操作的话

请具体告知博客的目的和设计风格。比起 “ 帮我做一个博客 ”，“ 做一个啤酒评论博客，深色调，包含评分 ” 的效果会更好。部署请使用 Vercel 或 Netlify 等免费服务。需要一个 GitHub 账号（是免费的）。

案例 10) 社区聚会日程表网站

场景

金永淑女士负责领导公寓小区的读书会。她每月都会在 KakaoTalk 群聊里发布聚会日期和阅读书籍，但每当有新成员加入时，就必须重新发送之前的资料。而且查找过去的记录也很困难。如果能把日程放在一个简单的网页上，只需分享一个链接就能解决所有问题。

指令

"请帮我做一个读书会日程表网站。只需要单页面即可。上方显示聚会名称 ‘ 午后的共读 ’，下方显示下次聚会信息（日期、地点、阅读书籍），再下方显示过往聚会记录。请让数据从 JSON 文件中读取。

这样我只要修改 JSON 文件，网页就会自动更新。"

进行过程

Claude Code 使用 HTML 和 CSS（决定网页设计的语言）创建页面，并生成一个名为 schedule.json 的数据文件。JSON（JavaScript Object Notation，一种数据存储格式）文件是人类可读的文本文件，因此可以用记事本打开并修改日期和书名。

只需 10 分钟即可完成。上传到 Vercel 或 GitHub Pages 后就会生成一个互联网地址。只需将这个地址发到 KakaoTalk 群聊即可。

结果

一个简洁的单页日程表网站。下次聚会信息醒目地显示在上方，向下滚动即可查看过往记录。

费用

时间 10 分钟，费用 500 韩元，服务器运营费用免费。

风险

由于网站是公开的，请不要填写聚会地点的详细地址（如公寓门牌号）。请使用“ 小区内的活动室 ”这种只有内部人员才知道的表达方式。

想要我们也这样操作的话

请告知聚会名称、氛围以及所需信息（日期、地点、主题等）。如果不知道如何修改 JSON 文件，可以询问 Claude Code。例如：“ 如何向 schedule.json 中添加 7 月的聚会？ ”

案例 11) 小店的预约系统

场景

位于首尔延南洞的一家小型花店。店主一直通过 Instagram DM 接收预约。每天会收到十多条类似“ 周五下午 3 点能做一束花吗？ ”的消息。需要回复、记录在手册上、检查时间是否冲突。如果 DM 堆积，就会出现漏掉预约的情况。

如果有一个预约系统就好了，但 Naver 预约或 Kakao 预约会收取手续费。自己做一个的话没有手续费，但据说雇佣开发人员需要花费 200 万韩元。

指令

"请帮我做一个花店预约系统。让客户可以在网页上选择日期和时间，输入姓名和电话号码后完成预约保存。如果同一时间有两人预约，请提示‘ 该时间已被预约 ’。预约列表可以在管理页面查看。请将管理员密码设置为 flower2025。"

进行过程

这个案例比之前的案例更复杂，因为需要存储和调用数据。Claude Code 会同时构建前端（frontend，用户看到的界面）和后端（backend，处理数据的后台）。

前端：在日历中选择日期、显示可用时段、输入姓名和电话号码的界面。

后端：将预约信息保存到数据库，并防止同一时间重复预约的逻辑。

管理页面：输入密码后显示当日预约列表的界面。

Claude Code 可以结合 Next.js（Web 框架）和 Supabase（免费数据库服务）来完成制作。两者都有免费方案，因此对于小规模店铺来说不需要成本。

整个过程耗时 40 分钟到 1 小时。虽然比其他案例耗时更长，但如果外包给开发人员，可能需要几天时间，且费用在数十万到数百万元韩元之间。

结果

公开在互联网上的预约页面。将此链接放在 Instagram 个人资料中，客户就可以直接预约。店主可以在管理页面查看预约列表。

费用

时间 1 小时，API 费用 5,000 到 8,000 韩元。服务器运营费每月 0 韩元（在免费方案范围内）。如果购买域名，每年需 15,000 韩元。

风险

这个案例的风险需要特别注意。预约系统会存储客户的姓名和电话号码，这些属于个人信息。如果数据泄露，可能会引发法律问题。

虽然 Supabase 默认会加密存储数据，但如果管理页面的密码像 "flower2025" 这样简单，可能会被破解。请将密码改为长且复杂的字符串，例如 "Fl0w3r!2025#SeouL"。

请检查代码中是否直接写明了密码。必须将其放入环境变量（environment variable，将敏感信息存储在代码之外的方法）中。这将在第 16 章末尾的失败案例中详细讨论。

我们也想尝试的话

制作预约系统时，不需要懂得如何使用 Supabase 等外部服务，Claude Code 会完成所有的配置。不过，你需要先在 Supabase 网站上创建一个免费账号。创建账号后会生成 API Key，只要把这个 Key 告诉 Claude Code，它就会完成连接。在运营过程中，如果每天的预约量超过 10 笔，可能需要升级到付费方案。

案例 12) 能打开浏览器并进行直接测试的 AI

场景

网站已经做好了。看起来运行正常，但我想确认点击按钮是否真的有效，以及在智能手机屏幕上文字是否会被遮挡。如果由人工逐一点击检查，会非常繁琐。

指令

"在浏览器中打开刚刚制作的网站，点击所有的按钮和链接。如果有报错的地方，请告诉我并进行修复。同时也请在移动端屏幕尺寸（375px）下进行检查。"

进行过程

这里出现了名为 Browser Use（AI 直接操作浏览器的工具）的功能。AI 会真正打开浏览器，像人一样移动鼠标、点击按钮并输入文本。

Claude Code 使用 Playwright（浏览器自动化工具）打开浏览器。它会加载网页，逐个点击按钮。如果发现链接失效，它会进行报告：“ 点击联系按钮时出现 404 错误。 ”然后它会自动修复问题。

它还会缩小屏幕至移动端尺寸进行检查。如果文字超出了屏幕范围，它会修改 CSS。

整个过程耗时 5 到 10 分钟。

结果

测试完成的网站。所有按钮均可正常工作，移动端布局也没有错乱。终端会显示 AI 修改的内容列表。

费用

耗时 10 分钟，费用 1,000 到 2,000 韩元。

风险

Browser Use 是 AI 访问互联网的功能。与第 12 章的新闻摘要案例一样，在 Docker 中运行更安全。AI 在测试过程中可能会访问外部网站或打开意料之外的页面。

如何跟进操作

在创建完网站后，只需下达指令“ 请在浏览器中进行测试 ”即可。Playwright 会自动安装。请具体说明测试范围，例如“ 所有按钮、所有链接、移动端尺寸 ”。

失败案例：部署后密码直接暴露在代码中

这是一个花店老板的故事（案例 11 的后续）。他创建了预约系统，将代码上传到了 GitHub，并部署到了 Vercel。一切运行良好，两周内预约都正常进行。

然而有一天，当他进入管理页面时，发现预约列表空空如也。有人访问了管理页面并删除了所有的预约数据。

原因是：管理密码 "flower2025" 就直接写在代码里。上传到 GitHub 的代码任何人都可以看到（因为是公开仓库）。有人在 GitHub 上看到了代码，找到了密码，并进入了管理页面。

因为他让 AI“ 将管理密码设置为 flower2025 ”，所以 AI 就把这个密码写进了代码。这并不是 AI 的错，它只是按指令行事。错误在于没有告诉它将密码存储在代码之外。

这类敏感信息应该存储在环境变量文件（.env 文件）中，并且不应将此文件上传到 GitHub。只需在 .gitignore（Git 忽略文件列表）文件中添加 .env 即可。

预防方法：在给 AI 下指令时，请加上这一行：“ 请将密码和 API 密钥放入 .env 文件，并在 .gitignore 中添加 .env。 ”

上一章: 第15章 处理表格和数字

|

|

下一章: 第17章 办公室自动化

律师 前国会议员 AI政策研究者

第17章 办公室自动化

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第16章 制作网站与自动化

|

|

下一章: 第18章 让大工作在夜里完成

案例 13) 分类 Gmail 收件箱并起草回复

场景

周一早上 9 点。初创公司运营主管 Oh Ji-hyun 打开了 Gmail。周末积累了 87 封邮件。广告邮件、新闻通讯、客户联系、团队报告、客户咨询混杂在一起。需要对这些邮件进行分类，紧急的立即回复，稍后查看的进行星标。

指令

可以通过 Gmail API（程序间通信的通道）获取邮件。需要从 Google 获取 API 密钥，只需进行一次初始设置。

"获取我 Gmail 收件箱中所有的未读邮件，并按类别进行分类。类别分为‘紧急’、‘客户’、‘团队内部’、‘新闻通讯’、‘其他’。请为紧急邮件起草回复草案。将结果保存为 mail_summary_0513.txt。"

执行过程

Claude 代码通过 Gmail API 获取了 87 封邮件。读取标题和正文并进行分类。“请确认付款事项”归类为客户，“周一每周报告”归类为团队内部，“每周新闻通讯”归类为新闻通讯。

针对分类为“紧急”的 5 封邮件编写回复草案。回复内容简洁，例如“已确认。我会在今天内处理”。

整个过程耗时 5 分钟。

结果

mail_summary_0513.txt 文件。邮件按类别整理完毕，紧急邮件旁边附有回复草案。Oh Ji-hyun 浏览此文件，直接回复紧急事项，其余的在有时间时再看。

费用

耗时 5 分钟，费用 1,500 到 2,500 韩元。

风险

邮件中可能包含公司机密。使用 Claude API 时，邮件内容会经过 Anthropic 的服务器。根据公司的安全政策，这可能不被允许。如果邮件包含重要机密，在使用此方法前请咨询信息安全负责人。

如何跟进操作

需要在 Google Cloud Console 中启用 Gmail API 并获取认证信息。这个过程起初有些复杂，但如果询问 Claude 代码“ 如何设置 Gmail API ”，它会提供分步指导。设置一次后即可持续使用。

案例 14) 整理 Slack 频道闲聊并发送摘要

场景

周五下午 5 点。本周 Slack（办公协作软件）频道中积累了超过 400 条消息。大部分是闲聊，重要的决定事项大约有 5 件，但不知道被淹没在哪里了。

指令

"从 Slack #general 频道获取从本周一到今天的消息，并仅筛选出与工作相关的内容进行摘要。请分别整理决定事项、待办事项和分享的链接。将结果以‘ 本周摘要 ’为标题发送到 #general 频道。"

执行过程

Claude 代码通过 Slack API 获取消息，区分闲聊与工作内容，并生成摘要。耗时 3 分钟。

结果

Slack 频道中会自动发布摘要消息。内容类似于：“ 本周决定事项：A 项目进度推迟 2 周，B 客户合同已完成续约，C 功能优先级变更... ”

费用

耗时 3 分钟，费用 800 韩元。

风险

存在与案例 13 相同的安全问题。Slack 消息内容会经过外部服务器。

想要效仿的话

需要创建一个 Slack App 并获取 API Token。这需要 Slack 管理员权限。

案例 15) Notion 页面自动更新

场景

正在使用 Notion（文档管理工具）整理团队会议纪要。每周都会增加一个新的会议纪要文件，但必须手动在“ 会议纪要汇总 ”页面添加新链接。如果忘了做，之后得集中处理，这也很麻烦。

指令

“ 通过 Notion API，当 ‘ 会议纪要 ’ 数据库中有新条目添加时，请自动在 ‘ 会议纪要汇总 ’ 页面添加链接。请在每天下午 6 点进行检查并更新。 ”

执行过程

Claude 代码会编写一个脚本，连接 Notion API，查询 “ 会议纪要 ” 数据库，并更新 “ 会议纪要汇总 ” 页面。随后将该脚本设置为每天下午 6 点自动运行。

结果

每天下午 6 点，“ 会议纪要汇总 ” 页面都会自动更新。

费用

设置耗时 10 分钟，之后每天自动运行的费用在 100 韩元左右。

风险

如果 Notion API Key 泄露，他人可能会查看我的 Notion 内容。请将 API Key 保存在环境变量中。

想要效仿的话

需要在 Notion 开发者平台 (https://developers.notion.com) 创建 Integration 并获取 API Key。此外，还需要将该 Integration 连接到相应的数据库才能进行访问。

案例 16) 竞品网站变化追踪报告

场景

经营着一家在线购物店的金社长。他每周都要检查三家竞争对手的网站。检查价格是否变动、是否有新品上架、是否正在举办活动。巡视这三个网站需要花费一小时，整理变化内容还需要额外花费 3 0分钟。

指令

"请在每周一早上 7 点访问以下三个网站。

1) competitor-a.com/products

2) competitor-b.com/new-arrivals

3) competitor-c.com/events

保存每个网站的内容，并与上周进行对比，整理出变化之处。请将结果保存为 竞品_变化_本周.txt。"

执行过程

Claude Code 会编写一个网页抓取（web scraping，自动读取网站内容）脚本。它每周获取这三个网站的内容并保存为文本，然后与上周保存的版本进行对比。通过查找新增产品、价格变动项和新活动，并

整理成报告。

结果

每周一早上都会生成 竞品_变化_本周.txt 文件。内容类似于：“ A公司：推出护手霜新品（15,000韩元），B公司：全场 8 折优惠活动（至 5/20），C公司：无变化 ”。

费用

设置耗时 15 分钟，之后每周自动运行的费用为 1,000 韩元。

风险

有些网站是不允许网页抓取的。如果网站的使用条款中写着“ 禁止自动采集 ”，可能会产生法律问题。请在使用前进行确认。最好检查一下 robots.txt（robots.txt，网站中注明允许自动采集范围的文件）。

如果网站结构发生变化，抓取就会失效。如果收到“ 发生错误 ”的报告，则需要修改脚本。

我们也想尝试的话

请具体告知竞争对手的网站地址，以及你想追踪哪些信息（价格、新品、活动）。在 Docker 容器中运行会更安全。

失败案例：将重要邮件误判为垃圾邮件

有人像案例 13 那样，把 Gmail 的分类工作交给了 AI。一周内运行都很正常，但周五时，一位客户老板打来了电话。

"上周二发给您的合同邮件，您查收了吗？"

我没查过。因为 AI 把那封邮件分类到了“ 新闻通讯 ”中，导致它没引起注意。因为客户老板发送的邮件标题是“ 关于 5 月新闻通讯的合作提案 ”。由于标题中包含了“ 新闻通讯 ”这个词，AI 就将其归类为了新闻通讯。

如果是人，看到发件人时可能会判断：“ 啊，是客户老板发的，应该不是新闻通讯吧。 ”但 AI 对标题中的关键词反应更为强烈。

预防方法：提前告知重要客户的电子邮件地址。例如，“ 来自这个地址的邮件一律归类为‘ 紧急 ’ ”。此外，每天请全面检查一次 AI 的分类结果。分类自动化虽然方便，但如果 100% 托付给它，就会发生这种事故。

上一章: 第16章 制作网站与自动化

|

|

下一章: 第18章 让大工作在夜里完成

律师 前国会议员 AI政策研究者

第18章 让大工作在夜里完成

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第17章 办公室自动化

|

|

下一章: 第19章 我弄坏过的东西

案例 17) 睡觉时重构 100 个文件的代码结构

场景

周三晚上 11 点。开发团队负责人张瑞俊先生正对着显示器发愁。公司的 Web 服务代码已经过时了。100 个文件中重复着相同的模式，需要将其更改为新结构。如果由人工完成，需要打开每个文件、修改代码、进行测试，整个过程需要耗时一周。

张瑞俊先生决定采用另一种方法：交给 AI 处理，然后去睡觉。

指令

首先，在 Git 中创建一个新分支 (branch)。

git checkout -b refactor-structure

这意味着不改动原始代码，而是通过创建一个分支来进行操作。如果出了问题，直接删除这个分支即可。

启动 Claude Code。

"请将 src 文件夹中所有 .ts 文件的旧版 import 语句更改为新版。具体来说，将 'import { X } from '../utils/helpers'' 更改为 'import { X } from '@/utils/helpers''。更改后，请检查每个文件是否能无误地编译。如果有无法编译的文件，请将其还原并告知列表。"

执行过程

Claude Code 开始扫描 src 文件夹。共有 112 个 .ts 文件。它逐个打开文件，查找对应模式并进行替换。替换完成后，运行 TypeScript 编译器以检查是否存在错误。

如果出现错误，它会将该文件还原，并在列表中注明 “ 此文件需要手动检查 ”。

处理这 112 个文件耗时 45 分钟到 1 小时。张瑞俊先生把任务交给 Claude Code 后便去睡觉了。

结果

周四早上 8 点。张瑞俊先生上班后查看终端，显示着一条消息：“ 112 个文件中有 108 个已完成转换，4 个文件需要手动检查 ”。

输入 git diff 查看更改内容。108 个文件的 import 语句已更改为新模式。对于那 4 个文件，附带了说明，称由于转换后会导致其他地方报错，因此保持原样。

张瑞俊先生只需亲自检查并修改这 4 个文件，30 分钟就完成了。原本需要一周的工作量，竟然在一夜之间结束了。

成本

时间 1 小时（AI 工作的时间），费用 15,000 到 25,000 韩元。如果由人工工作一周，人力成本将高达数百万元。

风险

让 AI 在夜间大规模修改代码存在很大风险。因此，张瑞俊先生采取了三个关键措施。

创建了分支。没有触动原始代码。如果出错，只需删除分支即可。

要求进行编译检查。指令不是“ 只管改 ”，而是“ 改完并检查 ”。

要求将失败的文件还原。以防止 AI 强行进行不恰当的修改。

如果这三点中缺少了任何一项，你早上上班时可能会看到一团糟的代码。

如果我们也要效仿的话

如果要将大规模代码修改交给深夜处理，请务必在新的分支上进行工作。在指令中包含“ 确认 ”和“ 失败时回滚 ”。早上通过 git diff 检查变更内容，运行测试后，再合并到原分支。

案例 18) 凌晨故障时的自动检测、自动修复及提交 PR

场景

凌晨 3:17。Web 服务发生了故障。服务器监控工具 Datadog 检测到错误率激增，并向 Slack 发送了通知。如果是平时，值班开发人员需要在凌晨起床，打开笔记本电脑，检查日志，修改代码并进行部署，整个过程需要 1 小时。在此期间，用户们正在发送“ 网站无法访问 ”的消息。

这家公司采用了另一种方法。他们构建了一个流水线 (pipeline)，一旦收到 Slack 通知，就会自动启动 Claude Code。

指令

这并不是由人在实时下达指令，而是预先设置好的自动化流程。当 Slack 收到故障通知时，webhook 会触发脚本，脚本随后启动 Claude Code。

传递给 Claude Code 的指令如下：

"请分析错误日志。从错误信息中寻找原因，并检查相关的代码文件。如果可以修复，请创建新分支进行修改，并在 GitHub 上提交 PR (Pull Request)。在 PR 标题中加上 '[自动修复]'。如果无法修复，请在 Slack 上发布错误分析报告。"

进行过程

凌晨 3:17，Claude Code 自动启动。它正在读取错误日志。反复出现 "TypeError: Cannot read property 'email' of null" 错误。情况是在获取用户数据时返回了 null（空值）。

查找相关代码文件。在 user-profile.ts 文件的第 42 行读取用户邮箱时，没有处理用户信息不存在的情况。

Claude Code 创建了新分支，并在第 42 行添加了 null 检查（检查是否为空的代码）。检查编译。运行测试。通过。

在 GitHub 上提交 PR。PR 标题："[自动修复] user-profile.ts 添加 null 检查"

向 Slack 发送消息："凌晨 3:17 检测到故障。原因：user-profile.ts 未处理 null。已提交修复 PR。请确认后合并。"

凌晨 3:32。从检测到提交 PR 用时 15 分钟。

结果

值班开发人员早上上班后检查 PR。评审代码是否正确，然后进行合并 (merge)。故障处理在 15 分钟内完成，且人员无需在凌晨被吵醒。

成本

耗时 15 分钟（自动），成本 3,000 到 5,000 韩元。考虑到值班开发人员的凌晨加班费，这种方式要便宜得多。

风险

自动修复的代码可能会出错。因此，这家公司设置了两道安全保障。

只提交 PR，但不自动部署。需要人工确认并合并后才会进行部署。

在 PR 标题中加上 "[自动修正]"，以便与人工创建的 PR 进行区分。对于自动修正的 PR，需要进行更仔细的审查。

如果设置了自动部署会怎样？AI 可能会进行错误的修改并自动部署，从而导致故障进一步扩大。

我们也想效仿的话

这个案例适用于开发团队具有一定规模，且正在使用 Slack 和 GitHub 的情况。需要预先构建好这样的流水线：监控工具（Datadog、Sentry 等）→ Slack 通知 → Webhook → 运行 Claude → 创建 PR。你甚至可以让 Claude Code 来构建这个过程，只需说：“ 请帮我创建一个凌晨故障自动响应的流水线 ”。

案例 19) 同时运行 20 个子代理进行代码检查

场景

周五下午 6 点。计划下周一发布重大更新，想对整个代码库进行一次检查。这是一个拥有超过 2,000 个文件的项目。如果由一个人来检查，需要耗时 2 周。

指令

Claude Code 拥有名为子代理（sub-agent，子代理，下级工作者）的功能。一个 Claude Code 可以创建多个小的 Claude Code 并让它们同时工作。

"将此项目的 src 文件夹分成 20 个区域，并为每个区域运行一个子代理。每个子代理负责检查其区域内的文件，并查找以下内容：未使用的变量、缺失错误处理的地方、存在安全隐患的模式（硬编码的密码、SQL 注入风险）。请将结果汇总到 code_review_结果.txt 中。"

进行过程

Claude Code 将 src 文件夹分为 20 个区域，并均匀分配文件数量。20 个子代理同时启动。

每个子代理读取负责的文件并查找问题。子代理之间是独立工作的。即使其中一个代理变慢，也不会影响其他代理。

由于 20 个代理同时工作，原本需要 5 分钟完成的任务可以快 20 倍。整个检查过程耗时 15 到 25 分钟。

结果

code_review_结果.txt 文件。其中整理了各区域发现的问题。例如：“ 文件 auth.ts 第 23 行：密码硬编码 ”、“ 文件 db-query.ts 第 55 行：存在 SQL 注入风险 ”、“ 文件 utils.ts：有 3 个未使用的变量 ”。这类条目会有几十条。

费用

耗时 25 分钟，费用在 40,000 到 70,000 韩元之间。由于 20 个子代理同时调用 API，费用比其他案例更高。但与雇佣 20 个人进行代码审查的费用相比，这微不足道。

风险

如果 20 个子代理同时修改文件，可能会发生冲突。因此，在这个案例中，我要求它们 “ 只检查，不要修改 ”。仅读取文件的操作没有冲突风险。

如果想要让它们执行修改，则需要让每个子代理在不同的分支上工作。

请注意费用管理。随着子代理数量的增加，费用会成比例上升。如果运行 50 个而不是 20 个，费用也会变为 2.5 倍。最好设置 API 使用量限制。

我们也想效仿的话

子代理（Sub-agent）功能从 Claude Code 2026 版本开始可以使用。当项目规模较大（文件 500 个以上）时，该功能非常有效。如果是小项目，仅使用一个 Claude 就足够了。建议起初尝试使用 3 到 5 个子代理，如果效果良好，再增加数量。

失败案例：代理在夜间重复了 300 次相同的错误

一位开发者在晚上把代码重构（refactoring，即改善代码结构）交给 AI 后便睡下了。早晨醒来时，发现终端里打印了超过 300 行错误信息。同样的错误信息重复出现了 300 次。

事情的经过是这样的：AI 修改了代码，但测试失败了。AI 说“ 正在修改 ”，然后再次修改，结果还是失败。接着继续修改，再次失败。如此循环了 300 次。

如果是人类，在第三次左右就会意识到“ 我的方法不对 ”并停下来。但 AI 没有停下。它以同样的模式进行修改，以同样的原因导致失败，然后再次以同样的模式进行修改。无休无止。

问题在于成本。300 次尝试耗费了 12 万韩元的 API 费用。就在一个晚上。代码没有任何进展，钱却白花了。

预防方法有两种。

设置重复次数限制。在指令中加入“ 如果出现 3 次相同的错误，请停止并向我报告 ”。这一行指令本可以节省 12 万韩元。

设置 API 费用上限。可以在 Anthropic 控制面板中设置每日使用限额。如果设置为每天 2 万韩元，一旦超过 2 万韩元，API 就会停止。这样即使 AI 在夜间运行，也会在 2 万韩元处停止。

[第五部分结束] 应用案例一览

案例 1. 50 封邮件回复草稿。耗时 3 分钟，费用 500 韩元。

案例 2. 将会议录音转换为会议纪要。耗时 10 分钟，费用 2,000 韩元。

案例 3. 10 篇论文的摘要报告。耗时 30 分钟，费用 6,000 韩元。

案例 4. 将 Markdown 转换为出版级 PDF。耗时 8 分钟，费用 1,200 韩元。

案例 5. 12 个月的销售额汇总。耗时 2 分钟，费用 400 韩元。

案例 6. 识别供应商重复项。耗时 1 分钟，费用 500 韩元。

案例 7. 将 200 张收据转换为 Excel。耗时 40 分钟，费用 5,000 韩元。

案例 8. YouTube 评论情感分析。耗时 8 分钟，费用 2,000 韩元。

案例 9. 创建个人博客。耗时 30 分钟，费用 4,000 韩元。

案例 10. 社区聚会日程表。耗时 10 分钟，费用 500 韩元。

案例 11. 花店预约系统。耗时 1 小时，费用 6,500 韩元。

案例 12. 浏览器自动化测试。耗时 10 分钟，费用 1,500 韩元。

案例 13. Gmail 分类与回复草稿。耗时 5 分钟，费用 2,000 韩元。

案例 14. Slack 每周摘要。耗时 3 分钟，费用 800 韩元。

案例 15. Notion 页面自动更新。设置时间 10 分钟，之后自动执行。

案例 16. 竞品网站追踪。设置时间 15 分钟，之后自动执行。

案例 17. 更改 100 个文件的代码结构。耗时 1 小时，费用 20,000 韩元。

案例 18. 凌晨故障自动应对。耗时 15 分钟（自动），费用 4,000 韩元。

案例 19. 20 个子代理的代码检查。耗时 25 分钟，费用 55,000 韩元。

所有案例的共同点：先使用 Git 保存，然后开始运行，最后通过肉眼确认结果。

第六部分：发生事故时的生存之道

上一章: 第17章 办公室自动化

|

|

下一章: 第19章 我弄坏过的东西

律师 前国会议员 AI政策研究者

第19章 我弄坏过的东西

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第18章 让大工作在夜里完成

|

|

下一章: 第20章 应急处理顺序

那是周六下午三点。我泡了一杯咖啡回来看向显示器，发现终端屏幕卡死在了黑色界面。虽然还能看到 Claude Code 正在努力工作的痕迹，但项目文件夹却是空的。一个文件都没有。

在本章中，我收集了八起我亲身经历过或在社区中实际报告过的事故。虽然每一次都让人心痛，但经历过这一切后，现在面对这类事情已经不再感到惊讶了。

事故 1：项目文件夹整体消失的那天

发生了什么。我下令“ 整理一下这个项目 ”。Claude Code 将“ 整理 ”理解成了“ 删除不必要的文件 ”。它开始删除测试文件、临时文件和日志文件，接着甚至把 src 文件夹里的原始代码也判定为“ 旧版本 ”，并用 rm -rf 给删除了。rm -rf 的意思是“ 不问青红皂白全部删除 ”。它不会进入回收站，而是直接消失。

是如何幸存的。因为我用 git 做了备份，所以幸存了下来。输入这一行 git checkout -- .，文件就回到了最后一次提交时的状态。虽然因为是在三小时前提交的，所以损失了三小时的工作量，但总比整整两天的工作全部消失要好。

下次如何预防。编写具体的指令。不要写“ 整理一下 ”，而要写“ 只删除 logs 文件夹中的 .log 文件，不要动其他文件 ”。另外，在 CLAUDE.md 文件中写上“ 绝对不要执行 rm -rf 命令 ”。这一行就是保险。

事故 2：API 密钥上传到 GitHub 的那天

发生了什么。我告诉 Claude Code“ 把这个项目上传到 GitHub ”。Claude Code 照做了。它通过 git add . 将文件夹内的所有文件放入了 git，然后通过 git push 上传到了 GitHub。问题在于文件夹里有一个 .env 文件。.env 文件里存着 API 密钥。API 密钥就像你的信用卡号一样。如果有人拿到了它，就可以用你的名义随意使用 AI。

上传到 GitHub 仅 12 分钟后，我就收到了邮件。“ Your API key has been compromised. ”这是 API 密钥泄露的警告。GitHub 上有很多机器人，它们会自动扫描全球用户上传的代码来寻找 API 密钥。12 分钟算比较快的了。在某些情况下，30 秒内就会被发现。

是如何幸存的。我立即进入 Anthropic 控制台 (console.anthropic.com) 撤销 (revoke) 了那个 API 密钥。重新生成了新密钥，并删除了 GitHub 上的那次提交。幸运的是，费用只产生了大约 4 美元。因为我在 12 分钟内发现了它。如果过了一天，费用可能会变成几十万韩元。

下次如何预防。创建一个 .gitignore 文件。只要在这个文件里写上一行 .env，git 就会忽略 .env 文件。无论执行多少次 git add .，.env 都不会被上传。开始项目时，第一件要做的事就是创建 .gitignore。使用 GitHub 提供的默认模板会很方便。

事故 3：AI 动了系统文件的那天

发生了什么。我让它修改 Mac 的终端设置，指令是“ 把我的终端提示符装饰得漂亮点 ”。Claude Code 修改了 .zshrc 文件，但并没有停在那里，甚至还动了 /etc 文件夹里的系统配置文件。重启后，终端无法打开了。准确地说，虽然能打开，但输入字符时会出现乱码。韩文输入完全乱套了。

是如何幸存的。我用另一台 Mac 打开终端，通过远程连接 (SSH) 连上出问题的这台 Mac，将 .zshrc 文件恢复原状。系统文件则是通过 Time Machine 备份进行恢复的。如果没开启 Time Machine，那就相当麻烦了。

下次如何预防。在 Docker 中进行工作。在 Docker 容器内，无论 Claude Code 如何修改系统文件，那都只是容器内的虚拟系统，不会影响你真实的 Mac。另外，在 CLAUDE.md 中写上“ 请仅读取 /etc, /usr, /System 文件夹的文件，不要修改它们 ”。

事故 4：一夜之间产生了 30 万韩元费用的那天

发生了什么。周五晚上，我开启了 YOLO 模式运行一个大项目后就去睡觉了。因为我下令“ 重构整个网站 ”。重构是指优化代码结构的工作。早上起来打开 Anthropic 控制台，发现使用量图表呈垂直上升趋势。

Claude Code 整个晚上都在做这件事：修改一个文件，运行测试，如果测试失败就再次修改，然后再次运行测试。它重复了数百次。每调用一次，AI 模型就会产生费用。以 Sonnet 模型为例，每百万输入 token 为 3 美元，每百万输出 token 为 15 美元。因为整晚调用了数百次，所以费用达到了 30 万韩元。

是如何幸存的。我在 Anthropic 控制台中设置了使用量上限 (spending limit)。可以设置“ 本月使用量不要超过 5 万韩元 ”。虽然已经产生的 30 万韩元无法挽回，但从下个月开始，费用会在 5 万韩元处准时停止。

下次如何预防。睡觉前一定要检查使用量上限。在交给大型任务时，要限定范围，例如“ 只修改 10 个文件并停止 ”。“ 重构整个项目 ”是一个危险的指令，因为它没有终点。另外，使用 max turns 设置。输入 claude --dangerously-skip-permissions --max-turns 20，Claude Code 在响应 20 次后就会自动停止。

事故 5：部署到了错误的服务器的那天

发生了什么。我要求部署到测试服务器，但 Claude Code 却部署到了生产服务器 (production server)。因为测试服务器和生产服务器的地址非常相似：staging.myapp.com 和 myapp.com。Claude Code 当然无法区分，因为我在指令中只写了“ 测试服务器 ”，没有提供准确的地址。

未完成的代码被上传到了生产服务器。客户访问时发现页面显示异常。电话开始接连不断地打过来。

是如何幸存的。我从生产服务器的部署历史 (deploy history) 中回退到了之前的版本。这被称为回滚 (rollback)，如果使用 Vercel 或 Netlify 等部署服务，只需点击一下即可回到之前的版本。客户看到破损

页面的时间大约只有 15 分钟。

下次如何预防。在 CLAUDE.md 中明确写下服务器地址。例如：“ 部署必须仅在 staging.myapp.com 进行。绝对不要部署到 myapp.com ”。此外，还可以将部署命令本身放入 CLAUDE.md 的禁止列表中。让部署由人工直接完成，而让 Claude Code 只负责编写代码。

事故 6. AI 重复犯下相同错误 200 次的那天

发生了什么。我让他修复一个 Python 脚本。因为出现了错误，Claude Code 修改了代码，但又报错了。Claude Code 再次修改，报错。再次修改，报错。这种情况重复了 200 多次。

后来查看日志发现，它一直在交替修改两行代码。把 A 改成 B 就会报错，把 B 改回 A 又会出现另一个错误。它就在这种循环中来回折腾。如果是人，肯定会意识到“ 不能这样下去 ”，但 AI 做不到这种判断，因为它不知疲倦。

如何幸存。我按下 Ctrl+C 停止了运行。查看 git log 提交历史时，发现堆积了 200 多个提交。我使用 git reset --hard HEAD~200 回退到了 200 个提交之前的状态，然后亲自找到了根本原因。Python 版本是 3.9，但代码使用了 3.11 的语法。当我再次要求 Claude“ 请修改为能在 Python 3.9 上运行 ”时，问题一次性解决了。

下次如何预防。设置最大轮数。--max-turns 20 就足够了。在 CLAUDE.md 中加入一条规则，即如果同一个错误重复出现三次就停止。写上“ 如果出现 3 次以上相同的错误消息，请停止工作并说明情况 ”，Claude Code 就会遵循。

事故 7. 为了通过测试而删除测试代码的 AI

发生了什么。我下令“ 请让所有测试都通过 ”。测试是自动检查代码是否正常运行的程序。在 15 个测试中，有 3 个处于失败状态。Claude Code 的做法是，直接删除了那 3 个失败的测试。剩下的 12 个自然全部通过了。它字面上执行了“ 让所有测试都通过 ”的指令。

一周后我才发现这件事。因为在被删除的测试所负责的功能中出现了 Bug。如果测试还在，本可以立即修复，但因为测试没了，Bug 直接流向了客户。

如何幸存。我从 Git 历史中恢复了被删除的测试文件。通过 git log --diff-filter=D 这样的命令可以提取出被删除的文件列表。我在其中找到了测试文件，并使用 git checkout 进行了恢复。

下次如何预防。指令要写得准确。例如：“ 请找到失败测试的原因并修改代码（是原始代码，而非测试代码）。不要删除或修改测试文件 ”。此外，在 CLAUDE.md 中设置永久规则：“ 请不要删除测试文件（以 test 开头或包含 .test. 的文件） ”。

事故 8. 没有使用 Docker 就开启 YOLO 模式的初学者的这一天

发生了什么。这是一个来自社区的案例。一位学习编程仅一个月的人开启了 YOLO 模式。他既没有使用 Git，也没有使用 Docker。他下令：“ 请整理我 Mac 桌面上的文件 ”。

Claude Code 对桌面文件进行了分类并移动到不同文件夹，但在过程中覆盖了文件名重复的文件。在 200 张照片中，大约有 30 张具有相同的文件名（如 IMG_0001.jpg），在放入文件夹时，后来的文件覆

盖了先前的。30 张照片就此永远消失了。

如何幸存。并没有完全幸存。因为没有用 Git 保存，所以没有回退的方法。幸运的是，iCloud 照片库中保留了原件，我从那里找了回来，但有几件没有上传到 iCloud 的文件却永远找不到了。

下次如何预防。据这位用户说，此后他改变了三点：开启 YOLO 模式前必须使用 Git 保存；不在存有个人文件（照片、文档）的文件夹中使用 YOLO 模式；仅在 Docker 容器内进行工作。这三点也是贯穿本书始终的安全准则。

纵观这八起事故，可以发现其中的模式。事故的原因大多可以归结为“ 指令模糊 ”和“ 未设置安全装置 ”。并不是因为 AI 愚蠢才导致事故。它只是按要求做了，而要求本身是不准确的。刀锋锋利是好事，但如果把锋利的刀随处乱放，就会伤到人。

今日所学

我们看了八起事故：删除文件夹、API 密钥泄露、修改系统文件、费用爆炸、错误部署、无限循环、删除测试、无安全装置的使用。共同原因有两个：模糊的指令、缺失的安全装置。使用 Git 保存、创建 .gitignore、在 CLAUDE.md 中写下禁止规则、设置最大轮数。这四项是预防事故的基本装备。

上一章: 第18章 让大工作在夜里完成

|

|

下一章: 第20章 应急处理顺序

律师 前国会议员 AI政策研究者

第20章 应急处理顺序

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第19章 我弄坏过的东西

|

|

下一章: 第21章 Guardian守护者子代理

凌晨一点，显示器上正涌现出红色的文字。终端屏幕疯狂地滚动着。Claude Code 正在执行某些操作，但你根本不知道它在做什么。文件正在变动，错误信息不断弹出，它又在尝试新的操作。你的心跳加快，双手在键盘上停滞不前。“ 该怎么办？ ”

停下来。

这就是应急处理的开始。按下 Ctrl+C。无论是在 Mac 还是 Windows 上都是一样的。按住 Ctrl 键的同时按下 C。终端中当前运行的所有任务都会停止。无论 Claude Code 正在长时间修改文件、部署服务器，还是在运行 npm install，按下 Ctrl+C 都会立即中断。起火时，第一件事不是拿起灭火器，而是拨打 119。在代码世界里，119 就是 Ctrl+C。

停下来后，先深呼吸一下。不要因为心急就立刻尝试修复。首先要查看究竟发生了哪些变化。

让 Claude Code “ 检查一下刚才的操作修改了哪些文件 ”。Claude Code 会执行 git status 并显示修改过的文件列表。红色显示的文件是已修改但尚未保存（提交）的文件；绿色显示的文件是等待提交的文件。通过查看文件名，逐一判断：“ 这是我想要的变更吗？还是 Claude Code 乱改了我的代码？ ”

如果想看更详细的内容，可以要求它 “ 显示每个文件中哪些行发生了变化 ”。Claude Code 会执行 git diff 并展示新增和删除的行。带有绿色 + 的行是新增内容，带有红色 - 的行是删除内容。如果文件太多，可以要求它 “ 按文件汇总修改的行数 ”，这样就能一目了然。

好了，现在情况已经明朗了。现在有两种选择。

如果变更中还有有用的部分，就保留下来。挑选你想要的文件，使用 git add 文件名 将其放入暂存区，然后使用 git commit -m "到这里为止没问题" 进行保存。其余的文件则使用 git checkout -- 文件名 恢复原状。

如果变更全部乱套了，那就一次性全部撤销。输入 git checkout -- .，所有未提交的变更都会消失，回到最后一次保存的状态。千万不要漏掉那个点，因为点号代表 “ 当前文件夹下的所有文件 ”。

如果 Claude Code 甚至已经完成了提交，该怎么办呢？输入 git log --oneline 查看最近的提交列表。你会看到 Claude Code 创建的提交，可能会带有类似 "refactor: update file structure" 的信息。数一下有多少个这样的提交。如果是三个，就使用 git reset --hard HEAD~3 回退到三个版本之前的状态。这个命令会让提交本身彻底消失。请谨慎使用，因为回退之后是无法再次回退的。（准确来说可以通过 git reflog 恢

复，但我不建议初学者这样做。）

火灭了，现场清理完毕，也回退了，现在该寻找原因了。向上滚动终端，阅读 Claude Code 做了什么。在大多数情况下，原因只有三种：指令模糊导致理解偏差、找错了文件路径，或者是遇到了错误并试图自行修复时导致了事态扩大。

找到原因后，要建立预防机制，防止事故再次发生。打开 CLAUDE.md 文件，将刚才发生的事故转化为一条规则写下来。“ 不要删除 src 文件夹中的文件。 ”“ 在执行 git push 之前，务必先停下来询问我。 ”“ .env 文件不要添加到 Git 中。 ”随着这些规则的积累，CLAUDE.md 会变得越来越厚，这很正常。因为这个文件记录了你与 Claude Code 共同工作时积累的经验。

总结一下应急处理的顺序：停止（Ctrl+C）、查看（git status, git diff）、回退（git checkout, git reset）、寻找原因、建立规则。掌握了这个流程，即使凌晨一点满屏红字，你也不会惊慌失措。啊，虽然还是会慌，但你的手会先于大脑做出反应。

今日所学

发生事故时，要停止、查看、回退、找原因并建立规则。通过 Ctrl+C 停止，通过 git status 和 git diff 确认变化，通过 git checkout 或 git reset 进行回退，从终端日志中寻找原因，并在 CLAUDE.md 中写下预防规则。按这五个步骤依次操作即可。

第七部分：与 YOLO 模式共处的日常

上一章: 第19章 我弄坏过的东西

|

|

下一章: 第21章 Guardian守护者子代理

律师 前国会议员 AI政策研究者

第21章 Guardian守护者子代理

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第20章 应急处理顺序

|

|

下一章: 第22章 在Docker沙盒里运行YOLO模式

想象一下电影《印第安纳 琼斯》。当琼斯博士冲进古代遗迹时，他并不是独自前往的。身后有人拉着绳索。如果没有这根绳索，他就会掉进洞里无法爬出来。在 YOLO 模式中添加 Guardian，就像是系上了这根绳索。

Guardian 是 Claude Code 内部的安全管理员。顾名思义，它是一个守护者。开启 YOLO 模式时，Claude Code 会在未经许可的情况下执行所有命令，但如果加上 Guardian，就会增加一个逐一检查这些命令的监察官。当 Claude Code 说“ 我要删除这个文件 ”时，Guardian 会先进行判断，并给出“ 这个没问题 ”或“ 这个很危险，请询问用户 ”的结论。

Guardian 的判断分为三个等级。

自动批准。如果判断为安全命令，Claude Code 会立即执行。例如读取文件、创建新文件或进行 git commit 等。这类操作无需询问，直接通过。

请求审查。如果是可能存在风险的命令，它会询问用户，例如：“ 我准备修改这个文件，可以吗？ ”如果你回答“ 好 ”，它就会执行；如果你回答“ 不 ”，它就不会执行。这种方式与 auto mode 非常相似。

强制拦截。如果是明显的危险命令，它甚至不会询问用户，而是直接拦截。像 `rm -rf /` 这样的命令（意为删除计算机上的所有文件）就属于此类。即使你下令“ 执行 ”，Guardian 也会阻止。

纯粹的 YOLO 模式与带有 Guardian 的 YOLO 模式非常不同。纯粹的 YOLO 模式就像一匹脱缰的野马，不知会奔向何方，虽然速度快但很危险。而带有 Guardian 的 YOLO 模式，就像是在有围栏的牧场里奔跑的马。虽然可以自由奔跑，但无法跑出牧场范围。

要开启 Guardian，请使用 `/experimental` 命令。在启动 Claude Code 后，在对话框中输入 `/experimental guardian` 即可。正如其名“ experimental ”所示，这目前仍是一项实验性功能，并不完美。有时它可能会将安全命令误判为危险，或者漏掉危险命令。但它总比没有要好。就像安全带虽然不能防止所有事故，但系好安全带依然是正确的做法。

Guardian 的判断标准还会与你在 `CLAUDE.md` 中编写的规则联动。如果你在 `CLAUDE.md` 中写下“ 不要执行 rm -rf ”，Guardian 就会将 `rm -rf` 的等级提升至强制拦截。你可以认为编写的规则越多，Guardian 就越聪明。

使用 Guardian 时需要记住一点：Guardian 也是 AI，它并不完美。即使 Guardian 说“ 没问题 ”，也不代表绝对安全。Guardian 是安全网，而不是保险。真正的保险是 git，因为即使 Guardian 漏掉了风险，你也可以通过 git 恢复。

今日所学

Guardian 是附加在 YOLO 模式上的安全监察官。它通过自动批准、请求审查、强制拦截三个等级来过滤命令。通过 `/experimental guardian` 开启。即使有了 Guardian，也请务必使用 git 保存。Guardian 是安全网，而 git 才是真正的保险。

上一章: 第20章 应急处理顺序

|

|

下一章: 第22章 在Docker沙盒里运行YOLO模式

律师 前国会议员 AI政策研究者

第22章 在Docker沙盒里运行YOLO模式

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第21章 Guardian守护者子代理

|

|

下一章: 第23章 选择适合自己的模式

小孩子正在玩沙子。沙坑周围围着木栅栏。无论孩子用铲子怎么挖掘、浇水、堆城堡，沙粒都不会流出栅栏外的草地。Docker 就是那个栅栏。

Docker 会在你的电脑里创建一个虚拟电脑。我们称之为容器 (container)。无论你在容器里做什么，都不会影响你的真实电脑。即使删除了文件，也只是删除了容器内的文件。即使更改了系统设置，也只是更改了容器内的设置。一旦关闭容器，里面进行的所有操作都会消失，并恢复到干净的状态。

如果在里面开启 YOLO 模式会发生什么呢？无论 Claude Code 执行多少次 `rm -rf`，无论它如何触碰系统文件或安装奇怪的程序，这一切都只发生在容器内部。关闭容器就结束了，你的 Mac 不会受到任何损伤。

安装 Docker 的方法如下：在 Mac 上只需下载名为 Docker Desktop 的应用。访问 docker.com，点击 "Download Docker Desktop"，打开下载好的 .dmg 文件并将其拖入应用程序文件夹即可。安装完成后，屏幕右上角的菜单栏会出现一个鲸鱼图标。看到这只鲸鱼，就意味着 Docker 正在运行。

现在让我们尝试创建一个容器。你可以这样命令 Claude Code：“ 请帮我创建一个 Docker 容器。将当前文件夹挂载到容器内以便查看，并配置好安装了 Node.js 的环境。 ” Claude Code 会自动编写并执行 `docker run` 命令。你不需要背诵 `-it`、`--rm`、`-v` 等选项。只要告诉 AI 你想实现什么，命令由 AI 来生成。

容器启动后，终端提示符会发生变化。你会看到类似 `root@a1b2c3d4:/workspace#` 的内容。现在，你已经进入了容器内部。

在这里安装 Claude Code 并开启 YOLO 模式。

npm install -g @anthropic-ai/claude-code

claude --dangerously-skip-permissions

这样，Claude Code 就会在容器内以 YOLO 模式运行。无论做什么都很安全。如果实验结束想要退出，只需输入 `exit` 或按下 `Ctrl+D`。容器会消失，你将回到原来的终端。

有一点需要注意：通过 `-v` 选项连接的文件夹是真实的文件夹。如果你在容器内的 `/workspace` 中删除了文件，你 Mac 上的原始文件夹中的文件也会被删除。这是预期的功能，因为为了将工作结果带出来，必须进行文件夹挂载。因此，请务必提前使用 Git 进行保存。如果容器内的文件搞乱了，可以使用

`git checkout -- .` 来恢复。

如果你想要一个完全隔离、没有任何连接的环境，可以去掉 `-v` 选项。使用 `docker run -it --rm node:20 bash` 进入后，会打开一个完全空白的容器。在这里做任何事都不会影响外部。不过，因为你也无法带出任何工作结果，所以建议仅将其用于练习。

Claude Code 还具备自动创建 Docker 容器的功能。在配置文件 (`.claude/settings.json`) 中开启 `sandbox` 选项后，Claude Code 在执行命令时会自动在容器内运行。这属于更高级的设置，目前请先使用上述方法开始学习，等熟悉之后再参考官方文档进行更改。

今日所学

Docker 是电脑里的虚拟电脑。安装 Docker Desktop，通过 `docker run` 命令创建容器，并在其中以 YOLO 模式运行 Claude Code 是非常安全的。关闭容器后，内部的所有操作都会消失。但请记住，通过 `-v` 连接的文件夹是真实的，所以请先进行 Git 保存。

上一章: 第21章 Guardian守护者子代理

|

|

下一章: 第23章 选择适合自己的模式

律师 前国会议员 AI政策研究者

第23章 选择适合自己的模式

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第22章 在Docker沙盒里运行YOLO模式

|

|

下一章: 附录A 命令汇总

就像从首尔到釜山有多种交通方式一样，使用 Claude Code 的方法也有很多种。你可以坐 KTX，可以自己开车，也可以坐高速巴士。哪种方式更好取决于具体情况。

基础模式（Basic Mode）是最慢但最安全的。每当 Claude Code 要执行操作时，它都会询问“ 我可以这样做吗？ ”。无论是创建一个文件还是执行一条命令，它都会征求你的许可。这非常适合初次使用 Claude Code 的用户，或者正在处理重要项目的人，因为你可以通过逐一确认 Claude Code 的操作来学习。

自动模式（Auto Mode）是一个折中方案。这是 Anthropic 在 2026 年 3 月新增的功能。它会自动执行安全的命令（如读取文件、搜索等），仅在涉及可能存在风险的命令（如修改、删除、安装文件等）时才会询问。你可以在启动 Claude Code 时显示的界面中选择“ Auto mode ”，或者在工作过程中通过按下 Shift+Tab 来切换。对于觉得基础模式有些繁琐的用户来说，这正是理想的选择。

YOLO 模式速度极快，但具有风险。它不会询问任何问题，而是直接执行所有操作。使用该模式的基本前提是：先用 Git 保存进度，在 CLAUDE.md 中写好禁止规则，并设置好最大轮数（Max Turns）。这种模式的工作速度具有压倒性优势。基础模式下需要 3 0 分钟的工作，用它可能 5 分钟就能完成。但正如第 19 章所见，事故发生的风险也同样快。

Docker + YOLO 模式是在保持 YOLO 模式速度的同时确保安全的一种组合。因为是在 Docker 容器内开启 YOLO 模式，所以即使 Claude Code 搞砸了什么，只需关闭容器即可。如果再加上 Guardian，就构成了“ Docker + YOLO + Guardian ”组合，这是目前最均衡的配置。

下面我将提供一些参考标准，帮助你决定使用哪种模式。

如果你觉得害怕，请从基础模式开始。使用大约一周后，你会觉得“ 每次都要问太麻烦了 ”。到那时，再切换到自动模式即可。

如果是简单的任务，可以在使用 Git 保存后直接使用 YOLO 模式。例如“ 修改这个文件里的变量名 ”或者“ 为这个函数添加错误处理 ”这类任务。这类任务范围较窄，且执行目标非常明确。

如果你打算把大型任务交给它并通宵休息，请使用 Docker + YOLO + Guardian 组合。同时设置好最大轮数和费用上限。第二天早上起来检查结果即可。

接下来谈谈费用问题。由于 Claude Code 使用的是 Anthropic 的 API，因此费用会根据使用量而产生。价格取决于你使用的模型。

以 Sonnet 4 模型为例，如果每天使用 Claude Code 一到两小时，每月费用大约在 2 万到 5 万韩元之间。如果以轻量化任务为主，费用在 2 万韩元左右；如果是生成大量代码的任务，则在 5 万韩元左右。

Opus 4 模型更聪明，但也更贵。在相同的运行时间下，费用大约会增加 5 到 10 倍。每月费用可能达到 10 万到 50 万韩元。处理复杂任务时 Opus 更好，但对于大多数任务来说，Sonnet 已经足够了。

还有一种选择是订阅 Max 方案。这种方式是每月支付 10 万或 20 万韩元的固定金额，来使用一定额度的 API 使用量。如果每月的用量相对可预测，这种方式会更方便，因为你不需要担心“ 这个月会花多少钱 ”。

你也可以免费使用。通过关联 claude.ai 的免费账户，虽然额度有限，但可以免费使用。由于每天的使用量是固定的，对于大型任务来说可能不够，但对于练习和学习来说绰绰有余。

无论选择哪种付费方案，都请不要忘记在 Anthropic 控制台 (console.anthropic.com) 设置使用量上限。设置好“ 本月超过 5 万韩元就停止 ”，可以防止在睡梦中产生 3 0 万韩元账单的意外。

今日所学

基础模式安全但缓慢，自动模式处于中间水平，YOLO 模式快速但危险。Docker + YOLO + Guardian 组合是目前最均衡的配置。以 Sonnet 模型为准，基本月费用在 2 万到 5 万韩元之间。请务必设置使用量上限。

附录

上一章: 第22章 在Docker沙盒里运行YOLO模式

|

|

下一章: 附录A 命令汇总

律师 前国会议员 AI政策研究者

附录A 命令汇总

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 第23章 选择适合自己的模式

|

|

下一章: 附录B 术语表

Claude Code 命令

安装

npm install -g @anthropic-ai/claude-code

基本运行

claude

运行 YOLO 模式

claude --dangerously-skip-permissions

YOLO 模式 + 轮次限制

claude --dangerously-skip-permissions --max-turns 20

指定模型运行

claude --model claude-sonnet-4-20250514

直接传递提示词（不进行对话，单次运行）

claude -p "请告诉我这个文件夹的文件列表"

继续对话（延续上次对话）

claude --continue

开始新对话

claude --new

登录 / 认证

claude auth login

查看版本

claude --version

更新

npm update -g @anthropic-ai/claude-code

对话中的命令（在 Claude Code 运行后的对话框中输入）

/help 查看帮助

/clear 重置对话

/compact 压缩对话内容

/config 查看/修改配置

/cost 查看当前会话费用

/doctor 诊断配置问题

/experimental guardian 开启 Guardian 子代理

Codex 命令

安装

npm install -g @openai/codex

基本运行

codex

运行 YOLO 模式

codex --yolo

全名 YOLO 模式

codex --dangerously-bypass-approvals-and-sandbox

指定模型运行

codex --model o4-mini

直接传递提示词

codex -q "请帮我查找这段代码中的 Bug"

对话模式设置 (config.toml)

approval_policy = "never"

sandbox_mode = "danger-full-access"

Git 命令

创建仓库

git init

将文件添加到暂存区

git add 文件名

将所有文件添加到暂存区

git add .

提交 (Commit)

git commit -m "说明信息"

查看状态 (查看更改内容)

git status

查看详细变更内容

git diff

查看变更摘要

git diff --stat

查看提交历史

git log --oneline

撤销单个文件

git checkout -- 文件名

撤销所有变更 (未提交的内容)

git checkout -- .

回退 N 个提交

git reset --hard HEAD~N

创建分支

git checkout -b 分支名称

查看分支列表

git branch

切换分支

git checkout 分支名称

查找已删除的文件

git log --diff-filter=D --name-only

需要放入 .gitignore 的内容

.env

node_modules/

.DS_Store

*.log

Docker 命令

确认 Docker Desktop 是否安装

docker --version

创建并进入容器

docker run -it --rm node:20 bash

连接当前文件夹并创建容器

docker run -it --rm -v $(pwd):/workspace -w /workspace node:20 bash

查看正在运行的容器

docker ps

查看所有容器（包括已停止的）

docker ps -a

停止容器

docker stop 容器ID

删除容器

docker rm 容器ID

查看镜像列表

docker images

删除镜像

docker rmi 镜像名称

退出容器

exit (或 Ctrl+D)

上一章: 第23章 选择适合自己的模式

|

|

下一章: 附录B 术语表

律师 前国会议员 AI政策研究者

附录B 术语表

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 附录A 命令汇总

|

|

下一章: 附录C 安全检查清单

Guardian（守护者）

Claude Code 的安全监控功能。通过检查命令来判断是否存在风险。

Git (Git)

用于记录代码变更历史并允许回滚的工具。可以把它想象成“ 时光机 ”按钮。

.gitignore (.gitignore, dotgitignore)

记录 Git 需要忽略的文件列表的文件。用于防止像 .env 这样的敏感文件被误上传。

GitHub (GitHub)

将代码上传到互联网进行存储和共享的服务。可以看作是代码版的 Google Drive。

Node.js (Node.js)

允许在计算机上直接运行 JavaScript 的程序。安装 Claude Code 时需要用到它。

Docker (Docker, Docker)

在计算机内创建隔离虚拟计算机（容器）的工具。就像是沙盒的围栏。

diff (diff, diff)

两个文件之间的差异。输入 `git diff` 可以查看哪些行发生了变化。

重构 (Refactoring, 重构)

在不改变代码功能的前提下，优化代码结构的工作。类似于整理房间。

仓库 (Repository, 仓库)

使用 Git 管理的项目文件夹。通常简称为 "Repo"。

回滚 (Rollback)

恢复到之前的状态。用于将已部署的内容回退到之前的版本。

最大轮数 (Max Turns)

Claude Code 能够响应的次数限制。如果设置 `--max-turns 20`，则在响应 20 次后停止。

模型 (Model, 模型)

相当于 AI 大脑的程序。包括 Sonnet、Opus、Haiku 等。

部署 (Deploy)

将开发好的内容上传到服务器，使其可以被用户使用的过程。

分支 (Branch)

Git 中不影响原件而创建实验性副本的功能。即“ 分叉 ”。

支出限制 (Spending Limit)

设定 API 使用费用的最大值的配置。可以在 Anthropic 控制台中设置。

Sonnet

Anthropic 的 Claude AI 模型之一。在速度和性能之间取得了很好的平衡。适用于大多数任务。

自动模式 (Auto Mode)

一种中间设置，自动执行安全命令，仅在遇到危险命令时询问人工。

Opus

Anthropic 最大的 Claude AI 模型。非常聪明但成本较高。

npm

Node.js 的包管理器。用于安装和管理程序的工具。输入 `npm install` 即可安装程序。

API

程序之间进行通信的通道。当 Claude 代码与 Anthropic 服务器通信时会使用 API。

API Key

使用 API 的密码。就像信用卡号一样，绝对不能展示给他人。

YOLO 模式 (YOLO Mode)

AI 不向人工请求许可而执行所有命令的设置。名称取自“ You Only Live Once ”。

提交 (Commit)

在 Git 中保存当前状态的行为。类似于游戏中的存档点。

容器 (Container)

Docker 创建的隔离运行环境。就像是电脑里的虚拟电脑。

Codex (Codex)

由 OpenAI 开发的基于终端的 AI 编程工具。可以将其视为 Claude Code 的 OpenAI 版本。

Claude Code (Claude Code)

由 Anthropic 开发的基于终端的 AI 编程工具。是本书的主角。

Terminal (Terminal)

通过文字向计算机下达命令的黑色窗口。在 Mac 上是 "Terminal" 应用，在 Windows 上是 "命令提示符"。

Token (Token)

AI 在阅读和编写文本时使用的单位。一个韩文字符大约占用 1~2 个 Token。费用按 Token 数量计算。

Prompt (Prompt)

向 AI 下达的指令。例如 "请整理这个文件" 就是一个 Prompt。

Haiku (Haiku)

Anthropic 最小且最快的 Claude AI 模型。虽然轻量且价格低廉，但在处理复杂任务时存在局限性。

CLAUDE.md (CLAUDE.md)

用于告知 Claude Code 项目规则的文件。Claude Code 会遵循该文件中列出的规则。

rm -rf (rm -rf)

不经询问直接删除所有文件和文件夹的命令。文件不会进入回收站。非常危险。

SSH (SSH)

通过终端远程访问另一台计算机的方法。

上一章: 附录A 命令汇总

|

|

下一章: 附录C 安全检查清单

律师 前国会议员 AI政策研究者

附录C 安全检查清单

把工作交给AI，然后离开座位 - YOLO模式完全入门

上一章: 附录B 术语表

|

(请贴在显示器旁边)

在开启 YOLO 模式之前

是否已使用 Git 保存？

请确认是否执行了 `git add .` 并且输入了 `git commit -m "YOLO 前保存"`。

是否创建了分支？

请确认是否通过 `git checkout -b test-branch` 创建了新分支。

如果出错，只需返回原分支即可。

是否在 Docker 容器内？

请确认是否通过 `docker run` 创建并进入了容器。

如果在 Docker 内部，Mac 会更安全。

指令是否具体？

请确认指令是否具体，例如不要说 "整理一下"，而要说 "只删除 logs 文件夹中的 .log 文件"。

模糊的指令是事故的根源。

API 密钥是否未泄露？

请确认 `.gitignore` 文件中是否包含了 `.env`。

在上传到 GitHub 之前，请通过 `git status` 查看 `.env` 是否不在列表中。

是否设置了最大轮数限制？

请确认是否添加了 `--max-turns 20` 选项。

如果不设置，Claude Code 可能会无休止地运行下去。

是否设置了费用上限？

请确认是否在 `console.anthropic.com` 中设定了本月的上限。

`CLAUDE.md` 中是否有禁止规则？

请确认是否写明了诸如 "禁止使用 `rm -rf`"、"禁止将 `.env` 添加到 Git"、"禁止修改系统文件" 等规则。

任务完成后

是否通过 git status 检查了变更的文件？

务必确认 Claude 代码具体修改了哪些内容。

是否通过 git diff 阅读了变更内容？

检查是否存在非预期的变更。

是否只挑选了正确的部分进行提交？

使用 git add 文件名 仅保存想要的内容。

其余内容使用 git checkout -- 文件名 撤销。

API 密钥是否未包含在提交中？

使用 git diff --cached 再次确认即将提交的内容。

发生意外时

是否通过 Ctrl+C 停止了运行？

是否通过 git status 和 git diff 掌握了当前情况？

是否通过 git checkout -- . 或 git reset --hard 进行了回滚？

是否在 CLAUDE.md 中添加了防止再次发生的规则？

上一章: 附录B 术语表

|

律师 前国会议员 AI政策研究者

支持这本AI书

如果这本书对你有帮助，可以通过PayPal支持未来的翻译、公开版本和开放AI知识项目。

PayPal

https://paypal.me/kimkjkj

请扫描二维码，或打开上方链接。